White Paper Report:
Spacecraft Fault Management Workshop Results
for the
Science Mission Directorate,
Planetary Sciences Division

March 2009

Lead Author
Lorraine M. Fesq, Engineering Development Office,
California Institute of Technology,
Jet Propulsion Laboratory

Co-Authors and Workshop Organizers
George Cancro, Johns Hopkins University, Applied Physics Laboratory
Chris Jones, California Institute of Technology, Jet Propulsion Laboratory
Mitch Ingham, California Institute of Technology, Jet Propulsion Laboratory
Jesse Leitner, Goddard Space Flight Center
John McDougal, Marshall Space Flight Center
Marilyn Newhouse, CSC/Marshall Space Flight Center
Eric Rice, California Institute of Technology, Jet Propulsion Laboratory
David Watson, Johns Hopkins University, Applied Physics Laboratory
Julie Wertz, California Institute of Technology, Jet Propulsion Laboratory

Acknowledgements

The authors of this White Paper would like to thank Jim Adams, Deputy Director of the Planetary Science Division in the Science Mission Directorate at NASA Headquarters, for launching the organization of the Fault Management Workshop and for providing programmatic leadership and funding for this activity; Paul Gilbert, Discovery & New Frontiers Program Manager, as programmatic host of the Workshop; Ken Ledbetter, Office of the Chief Engineer at NASA Headquarters for his vision to instigate this workshop; and Lindley Johnson for his oversight and guidance. In particular, the authors thank the Steering Committee members — John McDougal (NASA Marshall Space Flight Center), Chris Jones (Caltech Jet Propulsion Laboratory), George Cancro (Johns Hopkins University Applied Physics Laboratory), Steven Scott (NASA Goddard Space Flight Center), and Raymond Whitley (NASA Goddard Space Flight Center) — for their continuing leadership and support in the planning and execution of the conference and the preparation of this White Paper.

Part of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Contents

I Executive Summary

The science objectives for modern, deep-space missions, and the anticipated goals for these missions in the near future, are driving new requirements for spacecraft fault management. Traditional safing approaches, however, might be insufficient or inappropriate for certain critical events. In these cases, onboard resources and control logic must be used to manage fault events. We define Fault Management (FM) as the ability of a system to detect, isolate, and mitigate events that impact, or have the potential to impact, nominal mission operations. Note that this capability might be distributed across flight and ground subsystems, impacting hardware, software, and mission operations designs.

Fault Management is a critical aspect of deep-space missions; recent experiences, however, have highlighted a need to provide a focused assessment of the current state of practice in this area. In particular, the NASA Science Mission Directorate (SMD), Planetary Science Division (PSD), has experienced a number of technical and programmatic issues related to FM on recent missions. As a result, SMD/PSD commissioned an invited workshop with participants from the government, industry, and academia to:

- Assess the state of the art in both practice and research.
- Identify current and potential issues.
- Make recommendations for addressing those issues.

The workshop was held April 14 - 16, 2008, and was attended by one hundred engineers, program managers, and researchers. Also, in preparation for the workshop, the workshop organizers conducted a detailed survey of FM practices in the SMD/PSD spacecraft development community. This white paper describes the objectives and conclusions of the workshop and survey, laying out a roadmap for both near- and long-term actions that could be taken to address SMD/PSD concerns.

The workshop was structured into multiple sessions that included formal presentations of current mission experiences and relevant research. In addition, a significant amount of time was spent in three focused discussion sessions that addressed particular aspects of the FM problem. Specifically:

- FM Architectures.
- FM Verification and Validation (V&V).
- FM Development Practices, Processes, and Tools.

The results from each of these sessions were presented in terms of "lessons learned," "best practices," and "opportunities for investment." In this white paper, we have combined the results from these three sessions with FM survey responses and information from the presentations of current mission experiences to create a single set of top-level findings and recommendations. The top-level findings and recommendations from the workshop are presented in Table 1.

Table 1. Top-level findings and recommendations from the spacecraft fault management workshop

#	Finding	Recommendation
1	Unexpected cost and schedule growth during final system integration and test are a result of underestimated Verification and Validation (V&V) complexity combined with late resource availability and staffing.	a) Allocate FM resources and staffing early, with appropriate schedule, resource scoping, allocation, and prioritizing. Schedule V&V time to capitalize on learning opportunity. b) Establish Hardware / software / "sequences" /operations function allocations within an architecture early to minimize downstream testing complexity. c) Engrain FM into the system architecture. FM should be "dyed into design" rather than "painted on."
2	Responsibility for FM currently is diffused throughout multiple organizations; unclear ownership leads to gaps, overlap and inconsistencies in FM design, implementation and validation.	a) Establish clear roles and responsibilities for FM engineering. b) Establish a process to train personnel to be FM engineers and establish or foster dedicated education programs in FM.
3	There is a lack of standard terminology of FM systems that causes problems in reviews and discussions.	Standardize FM terminology to avoid confusion and to provide a common vocabulary that can be used to design, implement and review FM systems.
4	There is insufficient formality in the documentation of FM designs and architectures, as well as a lack of principles to guide the processes.	a) Identify representation techniques to improve the design, implementation and review of FM systems. b) Establish a set of design guidelines to aid in FM design.
5	Metrics have not been established to evaluate the appropriateness or measure the lifecycle progress of FM systems.	a) Identify FM as a standard element of the system development process (e.g., separate WBS) to promote innovative solutions and realistic estimates of complexity, cost, schedule. b) Establish metrics and process specification with milestones that will allow proposal evaluators and project teams to assess the relevance, merits and progress of a particular FM approach.
6	a) Practices, processes, and tools for FM have not kept pace with the increasing complexity of mission requirements and with more capable spacecraft systems. b) Indications of potential spacecraft anomalies exist in test data, but are not always observed or not adjudicated.	a) Design for testability: Architectures should enable post-launch and post-test diagnosis. b) Examine all observed unexpected behavior. c) Implement continuous process improvement for FM lifecycle. d) Catalog and integrate existing FM analysis and development tools, to identify capability gaps in the current generation of tools, and to facilitate technology development to address these gaps.
7	The impact of mission-level requirements on FM complexity and V&V is not fully recognized.	Review and understand the impacts of mission-level requirements on FM complexity. FM designers should not suffer in silence, but should assess and elevate impacts to the appropriate levels of management.
8	a) FM architectures often contain complexity beyond what is defined by project specific definitions of faults and required fault tolerance. b) Increased FM architecture complexity leads to increased challenges during I&T and mission operations.	Assess the appropriateness of the FM architecture with respect to the scale and complexity of the mission, and the scope of the autonomy functions to be implemented within the architecture.

#	Finding	Recommendation
9	FM architecture development is subject to changing priorities toward cost and risk over the course of system development.	Define and establish risk tolerance as a mission-level requirement.
10	a) The bulk of existing FM systems (e.g., mission-specific monitors and responses) is not inheritable. Heritage, similarity and inheritance assumptions tend to underestimate budgeting for necessary V&V activities and review milestones.b) Current FM architectures do not support significant re-use.	Examine claims of FM inheritance during proposal evaluation phase to assess the impacts of mission differences.
11	Inadequate testbed resources is a significant schedule driver during V&V.	Develop high-fidelity simulations and hardware testbeds to comprehensively exercise the FM system prior to spacecraft-level testing.
12	Organizations have different and sometimes conflicting institutional goals and risk postures that drive designs, architectures and V&V plans in different directions, causing friction between customers and contractors.	Collect and coordinate FM assumptions, drivers, and implementation decisions into a single location that is available across NASA, APL and industry. Utilize this information to establish / foster dedicated education programs in FM.

In synthesizing these findings and associated recommendations into a unified roadmap for NASA SMD, we established three general areas of focus: standardization (e.g., terminology and process), technology/tools, and training/education. In order to address the challenges of next generation FM, it will be important for NASA to address all of these factors. Note, however, that some particular areas should be prioritized for the near term on the basis of their potential for maximum benefit at minimal cost. Using the results of the workshop, we have sketched out a timeline of recommendations that describes priorities and dependencies in FM advancement over the next several years and identifies maturation paths for the three focus areas.

NASA management can address some of the recommendations directly, for example, setting up a WBS for FM that can be used to facilitate estimating and tracking FM efforts. Other recommendations are more complex, and will require the investment of resources to further analyze issues and develop best practices, including tools and processes, and to capture these practices in training materials and handbooks. To this end, our highest priority is to establish a forum, possibly a working group, through which the issues uncovered in the workshop can continue to be addressed. In the near term, we recommend the working group focus on the following issues:

- Standardize FM terminology and identify FM representation techniques,
- Establish FM metrics and develop an FM architectures trade space,
- Establish cost/risk estimation techniques.

Although more work and discussion are required to achieve consensus in the FM community, the most significant contribution of the workshop and associated activities is the recognition of Fault Management as a critical, distinct element of the spacecraft engineering process and the strong will within NASA SMD to advance spacecraft FM as an engineering discipline.

II Introduction

This white paper documents the findings and recommendations from the NASA SMD/PSD Fault Management Workshop held in New Orleans on April 14 - 16, 2008. This white paper provides the reader with the background necessary to understand the issues identified at the workshop, and documents lessons learned and best practices to assist future NASA planetary missions when planning, architecting, designing, implementing, and testing the Fault Management (FM) capabilities of a deep-space system. The scope of this paper covers the motivation for the workshop, the activities and events that took place during the workshop, the lessons and practices that were captured, and the resulting recommendations that emerged from the workshop. The target audience for this white paper includes current and future FM practitioners, proposal evaluators, and program/project managers.

A. Motivation

In recent years, a number of planetary missions have shown an increase in FM issues during integration and test (I&T) and flight operations, along with associated schedule impacts and lifecycle cost increases. Some of the issues noted include:

- Changes to the FM design late in the life-cycle, often resulting in a ripple-effect of additional changes in other areas.
- Inadequate understanding of system-level FM testing.
- Inadequate estimation of system-level FM testing.
- Unexpected results during FM testing requiring additional time for resolution.
- Operational limitations or restrictions placed on the spacecraft based on how the system was tested (in order to "fly-as-you-test").

These issues appeared and were recognized during reviews in almost every mission sponsored by the Planetary Science Division (PSD) of NASA's Science Mission Directorate (SMD). The issues appeared regardless of the organizations involved and occurred in both in-house NASA-developed missions and contractor-developed missions. The resulting schedule impacts jeopardized the mission's readiness for launch, which often is a very hard deadline for planetary missions (different launch windows often have severe ramifications to the outcome of the mission). The resulting cost overruns impact NASA's ability to fund other missions.

Because of the pervasive nature of these issues, the Deputy Director of the PSD and the Chief Engineer of the SMD recognized that there were likely systemic problem(s) that could be found to be the root cause(s). They also recognized that the problem(s) could be technical and/or process oriented. To begin to address these issues, the Deputy Director of the PSD assembled a Steering Committee consisting of representatives from GSFC, MSFC, JPL, and APL, and directed the Chief Engineer of the Discovery and New Frontiers Program Office to plan and implement a Fault Management Workshop. The direction given was to improve predictability and manageability in the design, test, and operation of planetary spacecraft FM systems. The workshop would pull together FM subject matter experts from government, industry, and academia to discuss their experiences on low-Earth-orbiting and planetary missions and offer their perspective on solving these problems. To achieve a better understanding of the issues, the workshop would address questions like the following:

- How could the FM development and system-level testing processes be more predictable from a cost and schedule standpoint?
- What are the system-level design or lifecycle process aspects that drive FM changes late in the lifecycle?
- Are different FM approaches more or less susceptible to these issues?
- Are these issues occurring only on planetary missions or are similar issues happening on Earth-orbiters and/or in the human space flight program?

B. Workshop Goals and Scope

The goal of the workshop was to document key findings and make recommendations to benefit future missions by avoiding the issues expressed in the previous section. By capturing the lessons learned from past missions through an honest and open exchange and documenting best practices that have been used across these missions, the intent was to help current and future mission developers minimize FM design and testing issues and, thereby, control schedule overruns and cost impacts. The approach taken in organizing the workshop was to assemble key players in the spacecraft FM field across NASA, industry, and other organizations to:

- Capture the current state of FM.
- Expose the challenges associated with engineering and operating FM systems.
- Identify and describe the issues underlying these challenges.
- Discuss and document best practices and lessons learned in FM.
- Explore promising state-of-the-art technology and methodology solutions to identify potential investment targets.

The programmatic scope of the workshop focused on deep-space and planetary robotic missions since the observed challenges had all occurred on missions of this nature. However, workshop participants recognized that Earth-orbiting (EO) missions also suffered from similar symptoms, although perhaps to a lesser degree, and that there was sufficient overlap in FM architectures and V&V methodologies to warrant strong representation and participation from the EO community. The scope specifically did not include human-rated missions with the acknowledgement that these missions involved additional FM challenges that typically are not encountered on purely robotic missions. However, members of the human spaceflight community did attend with the goal of understanding the issues uncovered during the workshop and absorbing lessons learned and best practices that are relevant to their missions.

The technical scope of the workshop focused on the portion of the spacecraft that handles faults. For the purpose of the workshop and this white paper, we define Spacecraft Fault Management using NASA's ***Preferred Reliability Practices*** definition for Fault Protection:

> Fault Management (i.e., Fault Protection) = *"Fault protection is the use of cooperative design of flight and ground elements (including hardware, software, procedures, etc.) to detect and respond to perceived spacecraft faults."* *(NASA NO.PD-EC-1243).*

Spacecraft Fault Management (FM) is a critical aspect of deep-space missions. It provides the capability for the spacecraft to detect, isolate, and mitigate events that impact, or have the potential to impact, nominal mission operations. This capability might be distributed across flight and ground subsystems, impacting hardware, software, and mission operations designs. Fault Management often is identified using different terms, such as Fault Protection, Redundancy Management, Health Management, Fault Detection/Isolation/Recovery (FDIR), and Safing. In this white paper, we use the term Fault Management to capture and represent all of these terms and uses.

C. Workshop Activities

The NASA SMD/PSD Fault Management Workshop was held over 3 days (April 14 - 16, 2008) in New Orleans, Louisiana. Although the number of participants at the workshop was originally targeted for 50 - 60 attendees to promote an interactive environment, interest grew and, as a result, the final registration far exceeded the initial estimate: a total of 100 representatives from 31 organizations across government, industry, and academia (see Table 2). The attendees brought expertise derived from a wide spectrum of missions, in terms of operations, duration and size, and functional roles.

Table 2. Participants and Missions represented at the SMD/PSD FM Workshop

Institutions	
NASA	Ames Research Center, Goddard Space Flight Center, Headquarters, Jet Propulsion Laboratory, Johns
Other Government	Air Force Research Laboratory, Defense Advanced Research Projects Agency, Naval Research Laboratory
Academia	Carnegie Mellon University, Iowa State University, Massachusetts Institute of Technology, SRI
Industry	The Aerospace Corporation, AI Signal Research Inc., Ball Aerospace & Technologies Corporation, Bastion
Missions	
Low Earth Orbit	Global Precipitation Measurement, Hubble Space Telescope, TacSat, Tropical Rainfall Measuring Mission
Deep-Space Missions	Cassini, Dawn, Deep Impact, James Webb Space Telescope, Mars Reconnaissance Orbiter, Mars Exploration Rover, MESSENGER, New Horizons, STEREO
Other	Chandra X-ray Observatory, Constellation (Ares, Orion, Altair), Solar Dynamics Observatory
Functional Roles	
Engineers	Software reliability, spacecraft systems, software, technical supervisors, computer scientists, fault protection,
Managers	Program managers, V&V managers, flight system managers, section heads, group supervisors, division chief
Academia	Program director, professors

All attendees were expected to contribute to the workshop through presentations, posters, and/or active participation in the dialog during the breakout sessions. Participants were asked to identify technology issues and process issues that currently are driving unplanned cost growth and schedule growth in FM systems for unmanned, autonomous spacecraft. The workshop participants also were tasked with capturing best practices to address those issues, as well as opportunities for investment to mitigate or possibly even avoid the issues on future missions. The workshop was not looking to produce a recipe or a set of standards. Instead, the goal was to rise above institutional preferences and evaluate the applicability, strengths, and weaknesses associated with different approaches.

The workshop was organized around five components: (1) case study presentations, (2) Request for Workshop Input (RFWI), (3) targeted roundtable discussions, (4) invited speakers, and (5) poster presentations. A detailed description of the workshop format, including the agenda of presentations and breakout sessions, is provided in Appendix A. Current FM approaches and techniques were collected using the case study presentations and the RFWI responses. Thirteen case studies were presented that exposed issues dealing with in-flight anomalies, project FM flight experiences, project FM development experiences, and industry FM philosophy and approaches, as well as lessons learned from eight current or past missions. Attendees of the workshop were requested to provide responses to an RFWI describing the use of FM on projects at their institution. Breakout Sessions provided a forum for the targeted roundtable discussions to enable the participants to discuss the issues presented in the case studies on the previous day and to suggest additional issues that were relevant to the Workshop. The goal of the Breakout Sessions was to distill the information to uncover the root causes of the issues. Workshop sponsors invited three speakers from academia and one from the NASA community to present a different perspective on FM and some insight into future directions. Finally, the poster presentations provided an opportunity for the participants to explore emerging technologies and to discuss future opportunities for investments to improve fault management for future missions. Case study summaries, breakout session descriptions, invited speaker presentations, poster session abstracts, and the RFWI inputs are described in Appendices B through F, respectively.

III Workshop Results — Findings and Recommendations

During the Workshop, three key concepts emerged as central themes that are categorized as general observations.

First, the implementation of FM within the software domain is generally similar across NASA, Johns Hopkins University/Applied Physics Laboratory (JHU/APL), and industry, and can be described at a fundamental level as an "alarm-and-response" system. See "General Observations on FM Architectures" in Appendix C for details on the similarities and differences noted between the architectures from the participating organizations. In an "alarm-and-response" system, the software monitors information from various on-board sensors for conditions that are out of specified bounds and responds to violations by sending a sequence of commands designed to fix the problem. Low-level differences in software FM architectural implementation do occur, in particular in the areas of:

- How alarms and responses are represented and implemented.
- Whether alarms and responses are arranged hierarchically or in a flat structure.
- Whether responses can be single or multi-threaded.

Each difference represents a trade-off. For example, a multi-threaded approach ensures that the highest priority fault is dealt with first; however, it also allows responses to preempt other responses, which might lead to unexpected interactions that introduce new challenges when testing the system. Overall, the discussions within the groups increased overall understanding since participants could see the results of other organizations' trade-offs and implementations. More sharing of this type should be encouraged.

Second, there was general agreement that FM in current missions was not being limited by technology, but rather by a lack of emphasis and discipline in both engineering and programmatic dimensions. This is not to say that technology advancements related to FM are not indicated. Indeed, it was also generally acknowledged that current-generation technologies and approaches, such as rule-based systems, are not expected to scale up to meet the requirements of future deep-space missions. However, it is felt that the systems engineering problems of current-generation programs must be addressed to enable any real technology advancement in this area.

Third, the in-flight performance of the FM systems on the projects that were represented at the workshop was deemed successful. Among the respondents, FM design flaws have not had an impact on mission success, though some false trips have resulted in unnecessary Safing events. Some of the more complex systems did need a number of configuration changes. Most were attributed to deferred testing that uncovered errors during flight, but some reported needing updates in response to false trips.

The following section captures the key findings and recommendations extracted from the case study presentations, the RFWI responses, and the breakout session discussions. These findings are considered contributing factors to the issues identified in this white paper and introduce challenges when evaluating, designing, implementing, and testing FM systems. The recommendations were compiled by rolling up and consolidating inputs from all workshop participants. These recommendations were captured during the Breakout Sessions and reflect the diverse thought processes and opinions from the multiple organizations involved in the discussions. Authors and presenters were extremely frank when sharing their experiences, with the understanding that the sensitive nature of the original materials would be respected. Therefore, supporting data have been sanitized to preserve confidentiality.

FINDING 1 – Unexpected cost and schedule growth during final system integration and test are a result of underestimated Verification and Validation (V&V) complexity combined with late resource availability and staffing.

Perhaps the single most significant finding from FM development in recent mission experiences is that a lack of FM consideration in early mission phases is at least a partial cause of unplanned cost and schedule growth during system development. It is common in developing a project schedule to develop the most compact and compressed means to perform the range of functions needed to engineer, build, and test the system. Plans for V&V tend to represent a concise and fixed schedule that assumes that everything will proceed successfully, accommodating anomalies and failures in the V&V process through overall schedule margin. However, the tests, by design, should be pushing the system towards failure and seeing how it responds. The system-level I&T phase is an opportunity to understand and characterize how the spacecraft as a whole operates. Therefore, this critical phase of the spacecraft lifecycle should be a primary focus of resources, schedule, and staffing. Since system-level I&T occurs late in the lifecycle, programs often cut corners on one of the most critical engineering activities.

The most telling evidence that FM is not considered early enough in the design process is the large increases between planned labor and actual labor hours in recent missions. One case study, shown in Figure 1, gave an example of the FM task being planned for 0.5 Full-Time Equivalent (FTE) throughout the mission; in actuality, FM staffing peaked at more than 14 FTEs (during the system-level I&T stage). This was a common occurrence: FM is viewed initially as a side responsibility of a systems engineer, increases to a full time job as the mission progresses, and eventually requires an entire team is to deal with problems, testing, etc.

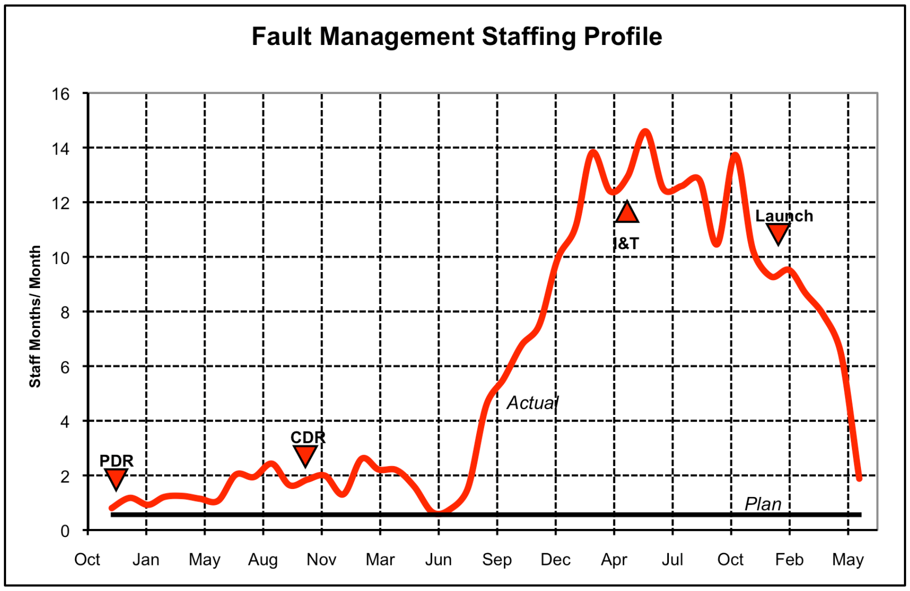

Figure 1. Planned vs. Actual Profile for FM staffing from a Case Study mission. This unplanned “bump” in staffing consistently appeared on numerous missions.

This unplanned bump in the staffing profile not only impacts budget, but also introduces logistical and efficiency challenges. People cannot start on a program and understand everything from the day they begin. Therefore, in these situations, the core team members need to spend time not only getting their job done, but also training the new people who are needed to handle the much larger workload than was anticipated.

Recommendation 1a: Allocate FM resources and staffing early, with appropriate schedule, resource scoping, allocation, and prioritizing. Schedule V&V time to capitalize on learning opportunity.

As space systems become more complex, it is becoming increasingly important that the process of designing the FM system begins in the very early stages of the mission design process; project management must also recognize that the FM system might require new architectural approaches. FM engineers should be involved during the proposal or initial planning phase. Also, a strong FM lead and team need to be supported throughout the program. Ground support equipment and personnel must be brought onto the project early enough to accommodate a ramp-up, training period typically required to be effective on a project. The message that "FM cannot be pushed back until the later stages of design" was reiterated throughout many of the case study presentations.

An overall FM strategy should be established early in system development, even as early as Pre-Phase A, to better understand cost and schedule requirements. This should result in a more efficient allocation of development resources later in system development. At a minimum, in the absence of a solution to the current staffing "bump" problem, the "bump" in the workforce profile must be anticipated and planned for from the initial planning stages. The project schedule and budget must include enough personnel and time for test program preparation and execution. Allowing no place in the project schedule for finding and fixing anomalies results in cut corners during testing. V&V planning should not be overly optimistic in its allocation of time and resources. This will help prevent slip in the V&V schedule and neglect of critical aspects of V&V.

FM requirements should be addressed in Pre-Phase A. This should be coupled with a top-level "testbed" strategy, also to be developed in Pre-Phase A. By Phase A, a FM plan and a standing FM Technical Interchange Meeting should be established. It is also recommended that a dedicated FM requirements review should be performed prior to system PDR. This review should cover testbed plans and make use of functional analysis and trade space analysis tools. A matrix of tool categories and potential/available tools was started during the workshop; an excerpt is shown in Table 3.

Table 3. Sample Matrix of Tool categories and availability for FM.

Purpose / explanation	Short Description	Representation Standard	Example	Comments
Fault containment impacts controller development	Operating System	N/A	VxWorks, ARINC 653, Honeywell, SAE standard…	
Fault reduction starts with good compilers (and coding practices)	Compiler	C, C++, Java, Lisp, Haskell, ML…		
Fault reduction via autocoding, sometimes directly from requirements	Software autocoder		MatrixX, Simulink, Labview, Autobayes,	MatrixX, Matlab and Simulink are high TRL; some autocoding magnifies

Almost all respondents to the RFWI identified common reviews as key milestones (e.g., Preliminary and Critical Design Reviews), but most also reported that additional reviews were helpful to address topics of specific interest. Several noted that a review of test capabilities and fault test plans was of particular significance to their projects. One respondent indicated that extensive changes were made to a project's test strategy, and test facilities were added in response to a review. Another participant opined that, while reviews are valuable, the formality of a FM systems review could impact its effectiveness; less formal, tabletop style reviews have better results than more formal reviews, where details can be obfuscated during the filtering process to generate formal slides.

V&V is a critical process that cannot be an afterthought for a project simply because it occurs late in the development cycle. System-level FM validation cannot be performed until late in the development cycle, as the system needs to be reasonably complete before FM systems can be fully tested. Moreover, the complexity of many FM systems makes them hard to validate. The V&V lead should be assigned during Phase A to question whether the FM design can be tested and to examine the design's complexity. In order to adequately characterize FM behavior, extensive testing is important, and review of tests by all of the system and sub-system designers is very important. However, the set of possible states and scenarios are too large to exhaustively test. Consequently, projects must take a measured approach to validation and must explore alternate techniques to complement hardware-in-the-loop testing.

All projects should have an incompressible test list of fault scenarios. Projects should apply protections so that FM scenario testing is not lost in the time crunch. Since FM testing is not currently deemed critical in developing a system, and testing typically occurs late in the V&V process, it can be tempting to eliminate testing of the range of key FM fault management scenarios. For mission assurance, protections should be established to mandate such testing.

Recommendation 1b: Establish Hardware / software / "sequences" /operations function allocations within an architecture early to minimize downstream testing complexity.

Key to the development of FM architectures is the understanding of the properties of elements employed to protect a system. Hardware, software, and operations procedures provide different levels of speed, isolation, reliability and flexibility to the design. The trade-off of what mechanism to employ and where to employ it is an important one. Unless this allocation is determined early in the project lifecycle, the FM architect will produce less optimal designs because implementation is forced into software, then into parameter-based autonomy, and finally into operational constraints, depending on how late FM architecture changes are applied. In addition, the FM requirements allocated to hardware or hard-coded software can take advantage of testing performed at the subsystem or hardware level rather than having all FM testing forced into system-level I&T.

Recommendation 1c: Engrain FM into the system architecture. FM should be "dyed into design" rather than "painted on."

It is a great benefit to flight projects for an architecture to ingrain itself within the early phases of design at the system level. The architecture should be intuitive to system engineers, subsystem leads, and operators such that a single design context can be carried from early design through to operations. In addition, the system design should be visible in the implementation.

A good FM architecture flows logically from mission requirements and the fault analysis of the system, including fault trees, probabilistic risk assessments, and failure modes and effects analyses. Early understanding of scope and a grounding in mission objectives and risks enable designers to produce an architecture that is focused on the most critical aspects of the mission or elements of the system. Developing the FM architectures from a systems point of view also offers the great benefit of reducing complexity of the overall design, since problems can be diagnosed with respect to overall system performance. In addition, responses can be developed in such a way as to reduce coupling to other elements of the FM architectures, reducing the complexity of system test efforts. Viewing the FM problem from a systems point of view also enables the design of generic protections that might remedy large classes of faults.

One of the invited speakers, Professor Brian Williams from MIT, provided insight into a unified framework to handle nominal and off-nominal scenarios within a complex system, such as a spacecraft. His presentation offered an alternate approach that captures the "dyed into the design" concept by integrating the management of faults into the baseline execution and control elements.

FINDING 2 – Responsibility for FM currently is diffused throughout multiple organizations; unclear ownership leads to gaps, overlap, and inconsistencies in FM design, implementation, and validation.

Participants have found inefficiencies in their definition of the FM engineering task. The lack of a strong process or clear relationships with other project tasks has given rise to a number of difficulties throughout the project lifecycle. In particular, a loose relationship with system engineering or late involvement of mission assurance in the FM development process has resulted in cost and schedule impacts and, in some cases, inadequate design. The lack of a clearly defined relationship between Systems Engineering and Safety and Mission Assurance organizations is problematic for FM design, implementation, and validation on flight projects.

Fault management tends to be viewed as a responsibility of system engineering. It is usually treated as a unique sub-discipline of system engineering and sometimes as a subsystem itself. Of the 11 missions that provided written input, 6 indicated that FM engineering was a responsibility of the project's systems engineering team; only one reported that FM engineering fell outside of systems engineering. (The remaining 4 missions did not provide information on their organization.) The design cycle of FM systems tends to follow a process similar to the rest of the project. However, the usual milestones frequently lag the rest of the system. For example, four respondents indicated that FM milestones follow the corresponding project-level milestones; others mentioned an augmentation of the FM review schedule that lagged behind the normal project reviews. FM engineering, however, is responsive to the detailed designs of subsystems, as failure analyses must be completed on the subsystems in order to complete the FM system design.

In developing the mission fault set, NASA centers tend to use a combination of failure analyses, mission assurance analyses, and subsystem engineer interviews. Practitioners in industry more commonly use heritage fault sets updated with subsystem interviews to converge on a fault set, with some use of failure modes and effects analyses when they are available. In all cases, interaction with subsystem engineers and mission assurance is paramount to assessing how a system can fail and defining the failure modes against which the system FM design will protect.

One respondent's input underscored the importance of a strong systems engineering approach to FM design. The project took a loose approach to the definition of its FM system — relying heavily on software architecture with no specific fault tolerance requirements, fault set, or definition of system behavior — and found that approach resulted in a system with too many flaws to be operable. Moreover, a lack of system requirements to verify meant that small flaws were discovered late, when testing was more expensive and bug fixes required significant effort. The organization has since taken steps to write strong FM requirements, perform detailed failure analyses to focus the implementation of FM, and follow a more rigorous test campaign starting at the unit level.

Recommendation 2a: Establish clear roles and responsibilities for FM engineering.

Clear roles and responsibilities need to be defined between Engineering and Safety and Mission Assurance organizations with regard to defining FM scenarios, identifying faults, and determining risk levels associated with identified faults. In particular, mission assurance should have a larger role in FM design.

While it is clear that a team responsible for the FM process needs to be put in place early, it is not clear what the organizational home of that team should be. Participants were in broad agreement that FM design is a system-level problem. However, there was not a consensus on whether FM responsibilities should fall to the existing systems engineering team or a separate team devoted to FM. Having a large core systems engineering team, without any specific FM engineers, means that FM will at times be pushed off for a higher priority systems issue. However, this team set-up would also ensure that the engineers designing the FM system would be experts in the overall system and could better understand the interactions and needs of the FM system. If FM engineering is a separate product team the effort is more focused, but it is separated from the design of the overall system, which makes the design process more difficult and might produce less robust fault protection. A third option is to keep FM part of system engineering, but designate a person or sub-team specifically responsible for FM. In this case, there could also be an integrated systems team, consisting of existing members of the systems engineering team focused on specific aspects of system design (fault management, flight software, testing, etc.), which meets on a regular basis to ensure that the system is integrating correctly.

One organization stressed that an integrated product team environment can provide a sound approach to ensure that a system-level perspective is enforced. System-level interactions can be the sources for a significant portion of on-orbit anomalies and failures. The integrated product team environment integrates all of the relevant skill sets throughout the project lifecycle. It can help ensure that all of the elements are brought together in the process and that component- and system-level behavior is properly characterized, tested, and stressed on the ground.

Ultimately, workshop participants universally agreed that, regardless of where the FM team is organizationally located, the team needs to be assembled at an earlier stage and the analysts (FMEAs, PRA) and the designers (FM, test engineers) need to have more interaction. Most participants agreed that FM engineering should be a system-level position on highly complex and/or critical missions.

Recommendation 2b: Establish a process to train personnel to be FM engineers and establish or foster dedicated education programs in FM.

Lessons learned from one project are easiest to apply to the next project if the personnel are the same. The more similar two missions are, the more benefit will be gained by having the same personnel (although the risk, of course, is that the architecture becomes stale without the influx of new ideas). Though deep-space missions tend to be unique — more so perhaps than “similarly repeated” Earth orbiters, even in unique missions — if the FM architectural approaches are slowly evolving, instead of being redeveloped from scratch for each mission, there will still be some knowledge base and lessons learned captured by using the same personnel on multiple missions. This concept could be extended to suggest the development of an evolving “set” of FM architectures, with different classes/types of mission requiring different architecture types (e.g., surface missions, flagship orbiters, flyby missions, etc.). Along these same lines, a FM process should be defined at each organization with team definitions and responsibilities laid out from the early planning stages.

Discussion at the workshop also addressed issues associated with organization and training. Rotation of engineering staff through different roles in FM (concept development, system development, mission assurance, operations) was cited as beneficial. In some organizations, alternation or rotation of staff between proposal and system development resulted in improved understanding and assumptions in FM development on subsequent missions. In larger enterprises, where dedicated FM organizations have been created, these organizations have proven to be good training ground for future mission system engineers. Unfortunately, the pipeline for (re-)staffing these organizations is not so clear.

The current generation of FM engineers typically is drawn from Attitude Control Subsystem/Guidance, Navigation and Control (ACS/GNC) or I&T organizations. Anecdotal information at the workshop indicated that these engineers are generally forced to learn on the job without formal training dedicated to the FM domain. Our recommendation is that NASA advocate for the creation of an FM academic/engineering discipline. Initially, this effort might focus on the development of training texts and special curricula (perhaps to include related application domains such as plant and automotive engineering). Depending on need, this might expand to dedicated FM engineering programs at key universities.

FINDING 3 – There is a lack of standard terminology of FM systems that causes problems in reviews and discussions, particularly when multiple organizations partner on a project.

Various organizations across NASA, JHU/APL, and industry use different terminology and different ways to represent a FM architecture. Terminology differences are so deep that there are even significant differences in the definitions of the terms used to describe the prevention and treatment of faults: fault management, fault protection, fault avoidance, redundancy management, or FDRIR (fault detection, response, isolation, and recovery). Representatives from various organizations are at times talking about the same thing and don't know it, and at other times think they're talking about the same thing and are actually talking about very different things. Table 4 provides a sample list of terms that often are used in the FM field, but are not universally defined.

Table 4. List of terms currently having multiple definitions when used within the field of Fault Management

Anomaly	Diagnosis	Failure
Fault	Fault management	Fault protection
Fault recovery	Fault tolerant	FDIR
FDRIR	Redundancy management	

One specific example of terminology differences across organizations was that there are two ways to interpret the term "single fault tolerant," as follows:

- In one organization's definition, a design was able to survive any single fault, and that fault could be an operator error, a hardware error, a software error, etc. Any combination of those faults was considered beyond single fault and outside the scope of the fault management system.
- A second organization viewed single fault tolerant as any single fault, hardware or software, but did not include errors such as operational errors or SEU induced errors, which are not considered faults.

In addition to causing difficulties in fault management design across organizations, terminology differences can cause problems within the same organization. At least one major spacecraft anomaly was partially caused by multiple organizations using similar terminology, but with slightly different definitions. This led to confusion, misunderstandings, and errors in design. This anomaly could have been avoided if the terms used in the design had been strictly defined and clearly stated to all team members.

Recommendation 3: Standardize FM terminology to avoid confusion and to provide a common vocabulary that can be used to design, implement, and review FM systems.

The first step in avoiding miscommunication and confusion during the design process and reviews is to create a standard terminology and to establish a well-defined set of metrics for FM characterization. Without standard terminology it will be difficult to organize FM into a discipline, and to address the other Findings in this white paper. Motivation and, perhaps, leadership for such an activity would properly come from NASA Headquarters; however, the activity should be performed by a broad spectrum of the community (such as that present at the workshop) in order to achieve overall consensus and acceptance.

Augmenting the recommendation of a common taxonomy (terminology) described above, it was felt that individual missions would benefit by creating a tailored version for the particular mission.

FINDING 4 – There is insufficient formality in the documentation of FM designs and architectures, as well as a lack of principles to guide the processes.

While using the same terminology will help communications among team members, another communications issue that was brought up several times was an inability to visualize these complex systems. Fault management systems have become so complex that visualizing all the monitors, responses, and where everything is getting done is becoming very difficult. It is also extremely difficult to visualize how a change in the system (monitor, response, or parameter) will affect the rest of the system. While visualizing the system is difficult for everyone, it is especially difficult for systems engineers who are not software specialists and who, as discussed earlier, are often the ones designing and implementing the monitors and responses.

The lack of visibility and understanding of FM design concepts translates into a lack of "reviewability." In some cases, FM designs are not expressed in a manner that makes them easily understood by developers or reviewers. All organizations share an inability to accurately describe the design of their FM architecture to system engineers and domain experts outside of fault management. In addition, examining the design or implementation of current FM systems does not readily reveal the system properties and overall concept of operations of the FM architecture. Lack of documentation results in ineffective project reviews and confusion when coordinating fault management with other subsystems and with system engineering. An important part of this problem is that little effort to date has been devoted to the application of rigorous architectural specification techniques and representations, such as those available in the broader engineering literature.

Finally, it was noted that unstructured knowledge transfer is a major FM cost driver. In current practice, FM requirements and design are captured at multiple points in the system development lifecycle, and transfer of that unstructured knowledge between mission phases can be a major cost driver.

Recommendation 4a: Identify representation techniques to improve the design, implementation and review of FM systems.

High-quality documentation of FM systems design is a boon to projects throughout the project lifecycle. A method of appropriately documenting and communicating FM architectures is needed to build stronger connections with other system elements and to provide for a better fault management solution by enabling more eyes on the problem. Documentation should address how the FM architecture meets institutional and Agency guidelines for fault management systems and should show justification for fault management design decisions. The documentation should also clearly address the allocation of functionality between hardware, flight software, ground software, and operations. Some process or tool that linked FM conceptual design, requirements generation/decomposition, design, development, I&T, and operation could result in major savings and risk reduction.

In addition, descriptive visualizations of FM designs are very important to success in all mission phases. Flow charts and clear descriptions of expected behaviors help reviewers, testers, and operators understand the system design in adequate detail to better fill their roles. Systems should be described in accessible terms. It is important to develop and use tools, such as diagrams, that help with visualization. When FM designers get too buried in FM- and software-specific language or representations, documentation becomes inaccessible to other reviewers and users. Finally, telemetry visualization tools — tools that show what happened, in what order, and what state the telemetry is in (red, yellow, green) — can also help operators to use these complex systems more efficiently.

Recommendation 4b: Establish a set of design guidelines to aid in FM design.

The case study presentations revealed several design guidelines that apply to all missions across the industry:

- Always provide a safety net, even if a specific fault has not been identified. Even if you haven't identified an exact failure mechanism, you need to design a way for the system to fail safe in every situation. Even when failure modes have been identified, the knowledge of those failure modes will always be limited and unexpected situations can occur. Therefore, it is important to put in mechanisms that will protect the spacecraft from known dangerous situations, even if a fault has not been specifically identified that can lead to that exact situation.
- The safing recovery procedure must be properly planned, tested, and practiced for rapid execution, if necessary. The safing recovery procedures should include identifying what data will be needed to characterize and recover from an unknown fault and must, therefore, be recorded and stored.
- Be careful of the limitations of redundancy. Performance on a backup system could be worse than performance on the prime system, even with a minor fault in the prime system. In addition, calibrating the backup could cost time and/or other resources. Therefore, even when redundant systems are available, it is often preferable to remain on a prime system with a minor flaw. Consequently, autonomous changes to redundant systems should be treated with caution.

While the complexity of future missions is driving the complexity of fault management systems, another design guideline that was mentioned several times was to keep the design as simple as possible and revert to fundamentals whenever possible. Although a high level of complexity is inherent in the FM of deep-space missions, whenever possible, the simplest design is often the best. An example was given of a very complex analysis that was finessed to increase science return. In the end, the analysis was incorrect and a spacecraft anomaly occurred. Whenever dealing with extremely complex problems, it is a good idea to make simplifying assumptions and to not forget the fundamentals of what the design is trying to achieve (see Recommendation #7).

Note: There was discussion of FM systems handling software failures and the emergence of software Failure Modes and Effects Analyses/Fault Tree Analyses (FMEA/FTA) efforts during one of the breakout sessions. This is a relatively new area for the aerospace community; although not widespread in practice, software FMEA/FTA analyses represent a potentially important opportunity to augment current FM design practices.

FINDING 5 – Metrics have not been established to evaluate the appropriateness or measure the lifecycle progress of FM systems.

Given the lack of standard processes, engineering roles, and terminology for FM, it is no surprise that there is a lack of standard metrics for evaluating the appropriateness of different canonical FM approaches to specific mission requirements and for tracking progress of FM development, test, and integration. In the course of discussions during the workshop, disagreements surfaced over the "best" approach to address the problem. Upon further reflection by the participants, it became apparent that many of these disagreements stemmed from the assumption that there was a single optimal solution to FM requirements across all mission classes when, in fact, the risks, constraints, and requirements across these classes implied fundamentally different approaches to FM. Our discussions included FM aspects of manned missions, deep-space missions, landed missions, and Earth-orbiting missions.

The only measures identified by workshop participants to describe FM processes and design were staffing levels (see Finding 1) and "rule counts" in architectures that support rules. While these might serve as rough measures of mission cost and complexity after the fact, there are no models for estimating mission cost and complexity on the basis of fundamental requirements or design parameters. It is a well-established fact that quantitative, measureable data are required not only for estimation purposes, but more fundamentally as a basis for improvement and maturation of engineering processes.

Recommendation 5a: Identify FM as a standard element of the system development process.

If NASA were to highlight the importance of FM by recognizing it as a separate and distinct element of the system development process, competitive forces in both research and development domains would produce new processes and technology to address many of the issues highlighted in the workshop. This recognition should include highlighting FM in program structure (e.g., dedicated Work Breakdown Structure [WBS] elements) and the use of generally accepted measures of risk, complexity, and performance.

One approach to addressing the problems of emphasis and discipline in new development programs would be for NASA to make FM engineering an explicit area of evaluation for competitive mission proposals. In the current environment, FM is not typically regarded as an engineering discipline, distinct from GNC, propulsion, communications, etc. Without this visibility, the cost and schedule impact associated with design, development, integration, and test of FM can be lost in early program planning, resulting in significant unplanned impacts late in the spacecraft development timeline. By emphasizing FM in all phases of the mission lifecycle as part of mission concept evaluation, NASA will create an environment where developers are motivated to produce innovative technical solutions and to correctly estimate the complexity of these solutions and the associated cost and schedule impacts.

Recommendation 5b: Establish metrics that will allow proposal evaluators and project teams to assess the relevance, merits, and progress of a particular FM approach.

A comprehensive suite of FM metrics should encompass risk, complexity, and performance. Performance could be further broken down to include "functional" measures, such as diagnostic coverage of the fault space, timing responsiveness of the fault responses, and determinism, and "non-functional" measures, such as testability, usability, and maintainability. Features that promote these properties should be inherent in FM architectures. Performance measures and progress metrics need to be specified for the key figures of merit, and reported during reviews and at major milestones of FM architecture development.

In the V&V phase, metrics should be employed in a number of ways. First, it will be critical to establish a complexity metric to assess the level of V&V effort needed. As expressed in Finding 7, the complexity of a system impacts how much resources are needed to perform system-level I&T for FM. Second, metrics are needed to measure progress in FM test suites. Not all requirements are created equal, and they should not be treated as such. Recommendation #6b provides additional suggestions for evaluating test results.

NASA can establish leadership in the areas recommended in 5a and 5b — namely a FM system development process and metrics — by commissioning a standards body composed of government, academic, and industry participants to create reference processes and metrics for FM. This will create a baseline for both evaluation of alternatives and tracking of progress during system development. Table 5 shows an example provided by one of the workshop organizations of a product that could come out of this standards body.

Table 5. Sample FM process specification.

Mission Phase	Focus	Tools
Conceptual Design (Pre-A)	• Understand Critical Risks and Associated Mitigation Costs	• Qualitative/Quantitative Risk Analysis
Preliminary Analysis (A)	• Preliminary Quantitative Risk Assessment • Architecture Trades • Cost Estimation	• FM Complexity Analysis • FM Costing
Definition (B)	• Probabilistic Risk Assessment • Preliminary FMEA • System & Subsystem Requirements Development • Testbed Plan	• PRA & FMEA Support • Test Planning
Design & Development (C/D)	• Formal Behavior Specification • Verification & Validation • Operations Training	• Logic Verification • Simulation • HIL Testbeds
Operations (E)	• Contingency Response • Degraded Operations	• Telemetry Analysis • Plan Generation • Operations Interface • Simulation & Testbeds

FINDING 6a – Practices, processes, and tools for FM have not kept pace with the increasing complexity of mission requirements and with more capable spacecraft systems.

Increasing capabilities in processors, sensors, and mechanisms are resulting in significant increases in the capabilities available to scientists for current and future missions. In order to employ these capabilities, engineers are using increasingly complex designs containing more subsystems with increasingly complex interfaces and embedded software. Instruments have become much more sophisticated. Vehicles are performing orders of magnitude more functions with each successive mission. As the systems become more complex, the number of potential fault scenarios for the system grows exponentially. While we put significant focus into developing the technology for complex spacecraft, the practices, processes, and tools for management of that complexity lag far behind. In particular, we don't have a way of evaluating or deciding that complexity has reached an unacceptable level of risk.

FINDING 6b - Indications of potential spacecraft anomalies exist in test data, but are not always observed or not adjudicated.

As spacecraft systems in general, and FM systems in particular, become more capable and more complex, test programs, especially system-level hardware/software test case, are becoming overwhelmed with data. Indications of problems are expressed during test, but are often not observed or are ignored. In several cases, reviewing test results after an anomaly showed indications of the problem, but the problem was not identified in the test because either a) the pass criteria of the test was met, b) inadequate analysis time was allocated, c) the test set was not comprehensive, and/or d) the schedule and budget pressures were forcing the test program forward.

In addition to indications of problems being missed in test programs, the indications of problems are also present, but not identified, in data during flight. While missions have telemetry sets with thousands of parameters, the parameters needed to diagnose a problem before it becomes an anomaly were, in some instances, not available in telemetry. This is often due to a lack of depth in the telemetry. In a situation very similar to test programs, telemetry often only alerts the ground once an anomaly occurs, but does not provide insight to the ground when the system is performing in an unpredicted way that does not officially fail the pass/fail criteria. A minimal design criterion for telemetry could be that the list of fault cases carried by the project can be detected and discriminated.

Recommendation 6a: Design for testability: Architectures should enable post-launch and post-test diagnosis.

In both test and flight programs, telemetry often only alerts the ground once an anomaly occurs, but does not provide insight to the ground when the system is performing in an unpredicted way that does not officially fail the pass/fail criteria. In both test programs and in telemetry definitions, we need to be looking for unexpected performance and early indicators of problems, instead of waiting until the thresholds for a given monitor are passed.

When recoverable faults happen in flight, the problem must be understood in a rapid manner to ensure spacecraft safety and understand any potential modifications to future operations. Architectures should enable rapid diagnosis (whether onboard or operator-in-the-loop) following a fault. Examples discussed during the workshop demonstrated the ability to save off data before and after a fault so that this data can be transmitted to ground tools that enable an operator to rapidly determine the root cause of a problem. Such a feature is also highly valuable during I&T, where data from system-level fault management testing can aid in post-test analysis and troubleshooting.

Recommendation 6b: Examine all observed unexpected behavior.

It is necessary to investigate all anomalies during a test program and in-flight. An unacceptable approach is to simply determine that the official pass/fail criteria have been met. Test programs need to be created to test against all aspects of expected performance, not simply to test to find complete failures. In both test programs and in telemetry definitions, we need to be looking for unexpected performance and early indicators of problems, instead of waiting until the thresholds for a given monitor are passed. Problems can be identified if the expected performance of the system does not match predictions in any area, including areas that are not being focused on in that particular test and including performance that does not satisfy fail criteria but is unexpected. A way to deal with complexity in system behavior is to attempt to define so-called “behavior envelopes” that redefine the testing/validation challenge to verifying that designed or even emergent behavior stays within a regime that is known to be safe, rather than testing all possible scenarios, which impossible for complex systems.

Recommendation 6c: Implement continuous process improvement for FM lifecycle.

The V&V process is, by nature, a learning process that should evolve throughout its duration. In a continuous fashion, lessons learned during the process should be used to update and improve the procedures for the remainder of testing. An appropriately architected V&V process should give rise to anomalies and failures on the ground. This is natural and desired because it is much less costly to deal with such issues on the ground. Workshop participants agreed that every anomaly or problem revealed during the V&V process should be fully addressed and resolved before launch to avoid occurrence on-orbit, where it would be costly and, perhaps, impossible to overcome. Emphasis should be on learning and applying from the V&V process vs. a goal that 100% of the planned tests are completed.

Solid pass/fail criteria should be established. Without clear definitions for pass and fail, it is too easy to call a test a success. Criteria must be pre-determined to ensure that test standards are maintained on the basis of actual testing requirements and on testing results. Additionally, proper tools should be developed to assess passage or failure of tests.

Recommendation 6d: Catalog and integrate existing FM analysis and development tools, to identify capability gaps in the current generation of tools and to facilitate technology development to address these gaps.

There is a strong relationship between tools and processes; therefore, one way to address issues in the FM development process is to establish a strong set of integrated tools based on the processes and measures described in Recommendation #5. A sampling of the FM tools in current use was collected at the workshop; this sampling is included in Appendix C, the summary of the Breakout Sessions. Although the list of tools is extensive, few, if any, are integrated or coordinated in terms of functionality or even terminology. One area where integration and coordination of tools might be particularly helpful is in the linking of analysis and design. In current practice, traditional analysis tools, such as FMEA, are not linked in any direct way with the tools and architectures being developed as part of system design. Linkage through common or integrated tools will facilitate communications and help eliminate unnecessary sources of error in the system development process.

Certain capability gaps are evident in the current generation of tools. During concept and requirements development, for example, tools for quantitative complexity analysis and complexity/risk tradeoff would be particularly valuable. These tools should be linked with costing tools to enable early, high-level decisions in cost/risk, hardware/software, and human/system trade spaces. Better tools for system behavioral modeling during early development were also cited as a current capability gap.

FINDING 7 – The impact of mission-level requirements on FM architecture complexity and V&V is not fully recognized.

Certain mission requirements were commonly cited as drivers for fault management designs. Common themes included the need for a fail-operational FM strategy, ground-in-the-loop response time, protection of resources, and the presence of a single fault-tolerance policy. Several responses also included discussion about how requirements were interpreted and applied. Almost universally, respondents were clear that the top-level mission requirements drive the complexity of FM systems most strongly, but that organizational cultures also have a notable influence.

The most stringent complexity driver appears to be the presence of a single fault-tolerance requirement. The requirement itself drives the design of a fully redundant system and a FM system that is capable of monitoring and managing that redundancy. Three missions that had significant trouble with complexity reported problems with interactions between fault monitors or responses., Three other architectures had explicit provisions to de-conflict fault management autonomy with other on-board autonomy. Other responses alluded to similar difficulties. In a rule-based architecture, the number of possible on-board response interactions increases as the square of the number of fault monitors and responses. Additionally, as a single fault tolerance requirement drives the number of fault cases that must be handled, single fault tolerance puts a lower bound on the system complexity necessary to meet the mission requirements.

Missions that require autonomous recovery of time-critical events tend to drive the project to a complex fail-operational strategy. Conversely, missions with less stringent performance needs are able to adopt a fail-safe approach. Every response collected mentioned a need to fail operational. In three cases, the mission did not require recovery of system-wide performance, but did identify certain functions that needed protection. Ion propulsion, for instance, requires a long-term thrust that, if interrupted for too long, might keep the mission from staying on the right trajectory. In other cases, thermal control, power production, or the protection of an instrument might impose tight pointing constraints, making reduced pointing modes impractical for vehicle safing. Similarly, strict availability requirements can also translate to fail-operational requirements when the ground-in-the-loop recovery time is too long.

The clearest driver for fail-operational designs is the need to complete a mission critical event. Six of the missions that provided input identified at least one event that required most of the system to fail operational. For interplanetary missions, encounters commonly come with some critical component, usually a timed event that, if not completed, could cause a loss of science or the mission: for example, a fly-by observation of a comet or asteroid requires stable, accurate pointing and a working science payload; orbit insertions or landings need propulsion systems or deployments to work reliably and on time. Functions that are necessary to completing these activities must be recovered in a timely manner to keep the spacecraft safe and to meet mission objectives.

Moreover, when performing a critical event at interplanetary distances, communications can have hours-long delays, which means most critical events are completed well before operators observe it on the ground. As such, the ground has little power to intervene in case of a fault, and mission designers must rely on autonomy to detect and correct faults in critical equipment.

Other conflicts over requirements have created unexpected problems for missions. Mission-specific requirements can impact system design, particularly when trying for re-use of an existing design. In one example, a requirement for communications coverage in safe mode forced changes to the existing telecom design. In another, very complex FM software was developed in response to the need for a flexible system that could be adapted for changing needs during different mission phases and deployed across multiple systems with different needs. The result is a complexity born of meeting differing or conflicting requirements in a single architecture.

Recommendation 7: Review and understand the impacts of mission-level requirements on FM complexity. FM designers should not suffer in silence, but should assess and elevate impacts to the appropriate levels of management.

There was discussion of particular design decisions that led to unnecessary complexity in FM design. It will be difficult and overbearing to provide a broad-brush set of fault management functions that apply to a wide array of missions. A project-specific fault management plan should be developed so as to avoid a costly V&V process definition that is burdened with management of non-credible faults, for example. One example cited was the use of concurrent fault-response logic, which greatly complicated integration and test. Strong consideration of the V&V impacts during mission requirements definition should be given.

In addition, critical behaviors should be implemented as simply as possible. Simple behaviors are easier to characterize and will execute more reliably. Critical behaviors like safing are too critical to a mission to add unneeded complexity. A recommendation is to limit the complexity of an FM system to only the necessary FM functions as defined by project-specific definitions of faults and required fault tolerance.

One approach to limiting complexity is to implement “dead-end” logic. If there is an appropriate end state for every response behavior, the emergent behaviors of the system are much easier to predict and significantly more stable. One project experienced the value of dead-end logic in flight when a repeating response would likely have been a risk to the mission.

FINDING 8a – FM architectures often contain complexity beyond what is defined by project-specific definitions of faults and required fault tolerance.

As a consequence of the intricacies and capabilities of fault management, FM systems often are required to protect and manage on-board resources. Seven of the 11 missions represented in written responses explicitly commented on some form of resource management in their FM systems. Because consumables are usually critical to mission life, FM implementation must protect them. Other resources, such as power, must be balanced to ensure the safety of equipment. FM systems must also manage any redundant assemblies that are brought on-line as part of a response in order to avoid conflicts. These examples of resource management needs can and do place non-trivial restrictions on the response to failures, often at the cost of increased complexity.

FINDING 8b – Increased FM architecture complexity leads to increased challenges during I&T and mission operations.

The more capabilities and complex operations we demand of a system, the more potential for faults we create and the more complex of a V&V process we require. A good number of difficulties in I&T are attributed to the complexity of fault management systems. At higher levels of integration, FM systems tend to have emergent behaviors that are difficult to predict, complicating the validation task. However, the problem is not limited to the most complex FM systems. During system test, one project found that slight variations in initial conditions resulted in different system behaviors. Such complexity not only was a difficulty in test, it also led to a large set of operational constraints that restricted the number of hardware configurations that could be flown once on-orbit. Many of the problems experienced that led to the discovery of large problems late could have been avoided with better early testing and more thorough behavioral analysis. In addition, there are indications that some FM architectures (e.g., rule-based systems) are difficult to scale up without introducing significant verification issues.

The flexibility and configurability of FM systems is one major contributor to increased testing complexity. It became clear in discussions that flexibility in systems is both a friend and a foe to the fault management engineer. Flexibility in an FM system is positive from a sustaining engineering point of view: it is essential for rapid modifications, which are often necessary to ‘tune’ these complex systems. FM architecture features that permit in-flight modifications are invaluable; workshop participants indicated that the ability to modify the fault protection on the fly saved several missions from mission failure. However, from the test perspective, it was also clear that flexibility in a system could be dangerous. This is especially true of ‘flat’ architectures that, when overextended, can result in emergent behavior that is hard to exhaustively capture in the V&V process. In addition, the response of any complex and flexible system can be sensitive to initial conditions and timing, making the system “fragile” and difficult to test. Even without additional complications from timing issues, N! permutations of a flexible system lead to N! test cases that need to be run. When the additional sensitivities to timing get added in (timing of faults can affect response paths), the test program quickly becomes too vast to cover every test exhaustively. This inability to exhaustively test the system has led to operational restrictions and adjustments on multiple missions. One mission in particular chose to restrict operations to only those states that have been tested. If a change in configuration (i.e., a swap to the redundant side) is required in-flight, the mission is put on hold until the tests related to that change are completed. It is clear that flexibility is important, but managing and restricting the flexibility to be used only when required is equally important. A related discussion is the trade between hardware-in-the-loop and simulation-based testing. The latter, in turn, implies the need for sufficient fidelity behavior models. These symptoms also hint at architectural weaknesses.

Recommendation 8: Assess the appropriateness of the FM architecture with respect to the scale and complexity of the mission and the scope of the autonomy functions to be implemented within the architecture.

Complexity in FM systems is a common problem: one of the root causes for not recognizing problems ahead of time is the complexity level of a fault management system. Some of the complexity grows naturally from the mission design and fault tolerance requirements, but additional complexity might be introduced for a number of reasons through the design process. FM designers must be aware of the consequences of complexity as it is introduced and must make deliberate decisions when introducing features that might increase system complexity. Non-essential sources of complexity, like configurability, must be considered carefully. Good engineering judgment needs to be used in system design, and the configurability of the system should not be an excuse to put off design decisions.

Flexible, configurable designs can reduce the work necessary to update the system for defects found late in the development process. Compiled code is significantly more costly to update than commandable parameters; however, the added flexibility does reduce the burden for regression testing. Alternately, the added complexity of a heavily parameterized system can make a system more difficult to analyze, test, and operate. In one example, the FM design even extended that flexible design to configurable responses, which proved to be too loose. In the end, the project chose to restrict what operators could do with that flexibility to maintain stable behavior. Configurable FM architectures should not be an excuse for loose implementations or delayed design decisions. Projects need to define a strong architecture and desired behaviors for FM systems early and keep a consistent approach throughout design and test. The overlapping concerns of resource management and fault management suggest that both should be addressed in a single architecture.

FINDING 9 – FM architecture development is subject to changing priorities toward cost and risk over the course of system development.

In the early mission phases, where the flight system architecture is being designed, the dominant project-level priority across multiple organizations was found to be cost. This can be clearly seen in staffing, where many organizations begin projects with the FM task staffed at 1 person or less. Late in the project lifecycle, however, organizations reported that the project-level priority switches from cost to risk, thereby forcing the fault management architecture to evolve to accommodate newly discovered risks to the project. Such architecture evolution is usually done in a one-by-one, ad-hoc fashion, and frequently occurs as a result of inadvertent mechanisms, such as requirements creep or shifting of organizational expectations. For example, one mission was designed with a high tolerance for risk, but eventually launched with a "must succeed" mentality that created additional stress on an FM strategy that was not in line with the risk posture of management. Other participants shared similar experiences.

Many organizations have responded to the changing priorities by introducing flexibility for late additions into their architectures. Other organizations preferred to focus on developing more "inherent" robustness in their architecture early in the program. Some advocated combinations of both adding robustness features and flexibility if used sparingly. Regardless of these options, all projects are helped by a tougher stance at the project level that considers overall risk in light of cost that will be incurred in I&T due to late changes. Moreover, the lack of ability to characterize this "incurred" cost is hampering project management decisions and potentially leading managers to lean to the risk side of the trade due to lack of impact information.

Recommendation 9: Define and establish risk tolerance as a mission-level requirement.

The consensus across all inputs that addressed these issues appears to be that 1) an early and clearly defined risk posture helps guide FM design and 2) consistency throughout the project is necessary to avoid friction and to settle on the best FM system for the project. Risk tolerance must be agreed upon and well defined early in a project's lifecycle.

The level of stress experienced in I&T is closely tied to how much risk can be tolerated and how much funding is allocated to testing. Therefore, gaining agreement from all stakeholders and lines of technical authority early in the project will help define the scope of testing.

Finally, project managers must develop and maintain a consistent risk posture over the life of the mission development. Drawing a clear line from risk posture to I&T cost will provide projects, mission managers, and reviewers with a better understanding of what is the end result of risk posture choices.

FINDING 10a – The bulk of existing FM systems (e.g., mission-specific monitors and responses) is not inheritable. Heritage, similarity, and inheritance assumptions tend to underestimate budgeting for necessary V&V activities and review milestones.

Claims of heritage that are not accurate can be cited as one of the main reasons that labor has been traditionally under budgeted for fault management. Budgets are often set too low for systems with re-use and heritage because it is assumed that the system will require a less extensive test program. Case studies have shown, however, that an extensive test program is still required for heritage systems. A new spacecraft with legacy software cannot be assumed to perform the same as a previous spacecraft, due to differences in timing and other minor system differences.

In addition to budgets being set too low in the case of actual heritage and re-use, budgets will also often get reduced based on false or exaggerated claims of heritage. In the proposal or planning stages, many missions will claim or assume a large amount of heritage in the fault management system, even when it is clear that major modifications will be required. When something is classified as heritage or "re-use" it often only has to go through peer-review, not a full design review. This tempts management into calling a fault management system heritage even when major adaptations might be required to meet the requirements of the new mission.

FINDING 10b – Current FM systems do not support significant re-use.

Many organizations employ engine-based or table-driven fault management system designs to facilitate software re-use. However, the core of the fault management functionality is not in the re-usable engine-based code, but rather the mission-specific alarms and responses that are designed and inserted into this code. Through the discussions of various types of FM architectures, it was clear that the inability to re-use any significant portion of the mission-specific design or implementation is a common problem across NASA, JHU/APL, and industry. The lack of re-use may be partially explained by the types of one-of-a-kind missions pursued by NASA.

A heritage system might not be appropriate for a new mission not only for technical reasons, but also because of cultural differences between partner institutions. Different organizations throughout the aerospace industry have varying fault management philosophies based on their organizational risk posture. Commercial organizations tend to have a slightly higher risk tolerance than government organizations; this is evident in their various fault management philosophies (See Finding 12). This difference in philosophies between various institutions means that for each mission, with different institutions providing different roles and responsibilities, a new mission-level philosophy needs to be developed. This unique compromise between various philosophies for each mission makes re-use of software extremely difficult.

Recommendation 10: Examine claims of FM inheritance during the proposal evaluation phase to assess the impacts of mission differences.

Within the fault management of any mission, logical patterns should exist that are applicable to future missions. Development of fault management re-use concepts, potentially similar to software patterns or objects, should be considered. However, the V&V engineer and project manager should look carefully at all of the components, environment, and modes of operation before exploiting heritage assumptions in an attempt to reduce the level of V&V performed. Use extreme caution when attempting or assessing claims of re-use of FM systems.

FINDING 11 – Inadequate testbed resources is a significant schedule driver during V&V.

The majority of test campaigns are implemented on test hardware that includes a mix of engineering-model hardware with simulated hardware and environments. Every project reported the use of multiple venues of varying fidelity. However, testbed resources often are insufficient, and availability is a source of contention during V&V. In one example, a project was pushed into the development of additional test resources by shifts in the interpretation of FM requirements.

The lack of adequate testbed resources for FM integration and test was cited by a number of organizations as particularly problematic in the later stages of system development. In many cases the I&T of FM logic must wait until all other aspects of system hardware and software are in place. Resources for early checkout of FM logic prior to or in addition to full-scale spacecraft testbeds would be extremely desirable.

Another major source of difficulty for multiple projects was the difficulty of operating test venues. In one example, a project experienced multiple problems with testbeds: lack of fidelity in a hardware simulation caused test failures that were later executed successfully on flight hardware, and the difficulty of developing test scripts limited the unit testing the project could do, pushing the discovery of small errors to later in the test program. Another project did not have an adequate simulation of a communication link that, if not discovered during an in-flight test, would have resulted in the loss of a mission element.

Recommendation 11: Develop high-fidelity simulations and hardware testbeds to comprehensively exercise the FM system prior to spacecraft-level testing.

Testbeds are a crucial element of the V&V process for FM and should, therefore, be given sufficient resources and be maintained in such a way as to provide the integrity necessary to perform pre-flight testing. The critical role of testbeds during the V&V process was discussed at length during the V&V Breakout Session. This recommendation is a collection of all of the testbed-related recommendations identified at the Workshop, including:

1. Testbeds must be kept up to date with the flight vehicle. While it is prudent to make use of testbeds developed for previous, similar vehicles, it is critical keep these testbeds up to date with the key parameters of the current vehicle.
2. The fidelity of each testbed and simulation should be clearly defined on the basis of mission phase. The credibility, likelihood, and impact of faults can vary tremendously between different mission phases. Henceforth, both the fidelity and the need for specific V&V tests should be defined on the basis of mission phase, not globally for the mission.
3. Stress testing is important and should be a required part of the V&V program. Stress testing in flight-like scenarios can help to ensure system robustness. This can significantly help to mitigate risk in single-string architectures or those with very limited redundancy.
4. Validation should be performed independently from the design developers. It is critical in the validation process to capture any inherent flaws or systemic errors in the design. This can only be assured through an independent process to avoid a repeat of the same flawed logic or erroneous analysis or assumptions.
5. Evaluate FM test suite quality by assessing test coverage across subsystems and mission phases. No single mission phase or subsystem is sufficient to characterize the vehicle's fault management.
6. Always consider operations when defining the I&T environment and flow. The operations define the boundaries to which the system should be tested. Consideration of the operations will help ensure that the testing limits are both necessary and sufficient to cover the environment to which the vehicle will be exposed in space. Operations should be brought into the V&V process early in the lifecycle to ensure that stress levels are appropriately defined.
7. Develop recovery procedures along with FM tests. The assumption from the start should be that failures will occur in FM testing and the procedures for recovery should be developed right along with the testing plans. This will ensure that anomalies and failures will be appropriately adjudicated at the time of testing.

FINDING 12 – Organizations have different and sometimes conflicting institutional goals and risk postures that drive designs, architectures, and V&V plans in different directions, causing friction between customers and contractors.

The decision and design process for FM architectures appears to be driven by different and sometimes conflicting factors across NASA, JHU/APL, and industry. The underlying reason for these differences is not considered explicitly when designing an FM system, and programs often lack the corporate knowledge or documentation of the rationale behind many early design decisions. The problem might be that the "why" is missing, so that it is difficult, if not impossible, to apply lessons to the next mission in anything other than a blind, knee-jerk fashion.

Participants articulated various different factors driving decisions and designs of FM architecture. These factors include institutional fears, heritage principles, heritage architectures, and high-level requirements on the timeliness of FM architecture to return a mission to normal operations. What is more intriguing is that some of these factors conflicted between institutions. For example, one organization's institutional avoidance of firing thrusters while out of ground contact directly conflicts with another organization's institutional avoidance of negative acquisition. What becomes apparent in this area is that little corporate knowledge or documentation of the rationale for these decisions exists.

Two respondents reported significant trouble with the interpretation of the single fault tolerance policy that created friction as projects implemented FM systems. The core problem these missions faced came down to a definition of what faults to protect against, something that varies across organizations. FM designers in industry will protect against the most likely failures, while designers within the NASA centers take a "possibility over probability" stance and design for all failures that might happen. When interpretation of the single-fault tolerance requirement is not clear early, a contractor will plan for a "probability" approach and subsequently be surprised to find that the NASA buyer expects a more rigorous "possibility" FM system. These conflicts were most prevalent in areas of the design where organizations with differing approaches attempted to share design responsibility.

Another symptom of conflicting assumptions and guidelines manifests itself during V&V. Most projects perform unit test on assemblies or modules as they become available, and higher-level verifications as the system is integrated on an engineering model or flight hardware as much as possible. Where organizations diverge is in the extent of high-level testing performed. Industry tends to focus on unit and integration testing and requirements verification. NASA, JPL, and JHU/APL go a step further and perform varying amounts of scenario testing for more rigorous validation of the system design. This difference is cited in a couple of instances as a source of friction, as one organization tends to expect more testing than another, but those expectations are not well-defined early.

The root of many of these challenges is the organizations' different positions on risk posture. A good example of this difference in risk posture was stated as the difference between designing against probability versus designing against possibility. These differences can manifest themselves during reviews, where institutional views drive assumptions on what is or should be the overall project risk posture. Though reviews are lauded as a great help to FM design, projects have found that overly zealous review findings have imposed unexpected FM requirements on projects and forced a shift in project risk posture that drove some of the problems during implementation and test.

Recommendation 12: Collect and coordinate FM assumptions, drivers, and implementation decisions into a single location that is available across NASA, APL, and industry. Utilize this information to establish / foster dedicated education programs in FM.

We recommend establishing a collection of requirements, driving factors, and implementation decisions in a single location to be made available to future fault management architects to provide a more complete view of the trade space and to enable more educated decisions for future projects. The culmination of this information would be an established vocabulary, suggested representation approaches, and a list of design principles utilized on prior missions. Each principle should be presented with an associated rationale statement, consequences (pro and con) of adopting the principle, and example implementations of this principle on past missions. Once developed, these materials could form the training material for educating future FM engineers (see Recommendation #2b).

IV Future Directions

A. Opportunities for Investment

A number of recommendations for emphasis or investment by NASA were discussed as part of the breakout sessions. These opportunities, summarized in the subsections that follow, are organized along the three breakout sessions of Architectures, V&V, and Practices/Processes/Tools.

1. Architectures Opportunities for Investment

The following opportunities for investment are derived from the Workshop discussions of lessons learned and best practices for FM architectures. These opportunities represent potential solutions to gaps identified in current fault management architecture practice.

- Capture existing FM architectures and requirements on mature programs. Collect design drivers and implementation decisions in a repository to provide a resource that enables future fault management architects to make better trades. Such a resource could also be used as a learning tool for new missions and young engineers.
- Develop and/or put into practice methodologies for more rigorous architecture specification to enable formal architecture-level analyses and facilitate architecture review and pattern re-use.
- Develop visual formalisms that facilitate FM architecture design and review, such that the architecture is understandable by system engineers and non-fault management domain experts.
- Articulate a comprehensive list of functional and non-functional properties for use as figures of merit in assessing FM architectures, and compile a mapping from architectural features to the functional and non-functional properties they promote (including examples of such features).
- Investigate architectures that inherently support rapid requirements-based testing early in the project lifecycle.

2. V&V Opportunities for Investment

The following opportunities for investment are derived from discussions to determine the lessons learned and best practices for spacecraft verification and validation. These opportunities represent potential solutions to gaps identified in the spacecraft V&V realm.

- Develop a means to confine complexity to testable units.
- Develop an approach to establish complexity containment regions.
- Develop an evolvable system model, capable of being validated by tests on flight hardware and software that is sufficient to be used for primary scenario and FM V&V.
- Develop a design environment/tool to capture desired system and FM behavior that is capable of dynamically executing the behavioral model.

- Develop a tool to choose which subset of tests to run when exhaustive testing is infeasible.
- Prioritize V&V actions with buy-in across the program.
- Develop, maintain, and update tools to support the V&V process. Include the following:
  - Tools to analyze test data in timely fashion.
  - Ops tools and ground tools.
  - Tools to capture code coverage accomplished during tests.
  - Configuration management tools for testing.
  - Design-time test generation tool.
  - A tool to highlight high water marks.
  - Tools to specify and monitor safety properties through development and test.
  - Success trees and fault trees.
  - Software simulation tools.

3. P/P/T Opportunities for Investment

The following opportunities for investment are derived from discussions to determine the lessons learned and best practices for FM Practices, Processes, and Tools. These can be grouped into "processes and tools" and "organization and training," each of which is summarized below.

a. P/P/T Processes and Tools

With regard to processes and tools:

- Processes and tools should be closely linked; however, at this point, it is apparent that more focus has been placed on the former in most organizations. Tool use was characterized as "viral" in nature, with good tools propagating between projects and organizations in an ad hoc manner, as opposed to being standardized and specified relative to a overall desired process and workflow in the FM development process.
- Tool integration should be facilitated through work on common terminology/taxonomy, metrics, and interface specifications. In particular, there should be work to integrate "top down" requirements development tools, such as fault tree analysis, with "bottoms up" design tools, such as FMEA.
- Complexity analysis tools should be developed for use in concept development and requirements definition. This would allow FM analysis to be incorporated into Pre-Phase A design centers, into mission costing models, and into various trade space evaluation processes. Ideally, tools would be available for behavioral modeling early in system design, and these tools would link with FM design, implementation, and test tools. Finally, process templates should be developed that build upon this new class of tools. It was felt that current knowledge across the community could be collected in an FM process template or handbook. Such as resource might specify different classes of mission with regard to their FM requirements and then define specific design guidelines for each.

b. Organization and Training

In the area of organization and training, there were only two top level recommendations:

- First, a recommendation for NASA to address the educational issues associated with establishment of a dedicated FM engineering discipline. This moves beyond the specification of a common taxonomy and set of metrics to include targeted university programs and texts.
- Second, a recommendation to begin bootstrapping a community dedicated to the engineering and science of deep-space mission FM. In some ways, this workshop might have served as a first step towards this goal.

B. Future Plans

Looking back on the original purpose for holding the workshop, we note the observation on multiple missions of an unplanned expenditure during spacecraft I&T to accommodate unforeseen or unplanned testing time for FM systems. To ameliorate this situation, the recommendations outlined in this white paper propose ways to make FM systems a) more predictable in development, in cost, and in schedule, and b) more manageable by identifying work units that are do-able by engineers at a specified level of experience. In this section, we identify a number of potential paths to follow to implement these recommendations. In addition, we strongly recommend the following activities:

- Hold subsequent workshops to identify solutions to the issues raised. It was beneficial to bring the community together to share ideas. The first workshop concentrated on assessing the current state and uncovering the common issues. The next workshop could focus on options and solutions and could include those disciplines that were weakly represented, such as Systems Engineering and Operations. Also, additional government and industry organizations should be included in these activities to expand the focus and to view what is being done in other industries.
- Establish a NASA Working Group for Fault Management that will take ownership of the issues identified, and establish ways to mitigate them within the NASA governance. This Working Group should be populated by all of the NASA Centers that are affected by the FM issues discussed in this white paper.
- Explore the following concepts:
  - How can the FM discipline achieve a more holistic view?
  - Do we need a paradigm shift?
  - Entertain ideas from outside of our field. What are the larger system management issues?

Table 6 provides a summary of a suggested implementation plan for the proposed timeframes for the opportunities for investment, displayed as an evolution of capabilities along a timeline showing when each could be accomplished. Table 7 suggests a potential roadmap that identifies maturation paths for multiple focus areas.

Table 6. Timeline of recommendations.

Focus Area	Near Term	Mid Term	Long Term
Taxonomy & Methodology Standardization	• Standard Lexicon • Fundamental Metrics • Standard Mission Types • FM Process Template(s)	• Refined Metrics & Performance Tracking • Process Standardization	
Technology & Tools	• Survey Current Tools • Survey Related Disciplines • Architecture Analysis • Complexity Analysis	• Design Specification & Review • Formal Methods for V&V • Cost/Risk Estimation	• Complexity Management
Training & Education	• NASA Training Courses • Coordinated Conferences	• Reference Handbook • University Programs	

Table 7. SMD/PSD planetary spacecraft fault management roadmap.

Thrusts	Explore	Plan	Develop	Deploy
Tools	Create trade space of existing	Identify tools to be	Create a complexity	
Architectures	Create trade space of existing			Technology
Processes	NASA-wide (or Center-wide or Directorate-wide) positions on FM: FM stance/template for different classes of missions; Design/evaluation criteria for FM systems; metrics specifications			
	Collect schedule and cost metrics of current processes to form bases of estimates -- e.g., staffing profiles through ATLO	Identify FM performance metrics -- reliability, coverage, operational availability, autonomy, complexity	Establish consensus on FM metrics/stance for different classes of missions.	Make FM an explicit effort within the standard flight project WBS and an explicit consideration as a proposal evaluation criteria
Training & Education	Courses: Identify existing courses within NASA and at universities that can be augmented with FM training			
	Presentations: present findings at workshops, conferences, NASA boards [e.g., QMSSW, Sandia Spaceborne Computing Workshop, Dx-08, FSW-08, 2009 IEEE Aerospace, 2009 InfoTech@Aerospace]			
	Textbooks: Review textbooks available today	Assess need: Textbook in work; Plan for a Practitioner's Handbook: Identify contents and authors		Publish Practitioner's Handbook
Standards	Terminology, Taxonomy - survey terms currently in use.	Establish multi-Center team	Establish common FM vocabulary, taxonomy	
	Representation - survey approaches currently in use.		Develop visual formalism that enables FM architecture design and review - e.g., SysML	

V Appendix A: Workshop Organization

A. Workshop Flyer

Fault management for today's deep-space missions is a complex problem, going well beyond the typical safing requirements of simpler missions. Recent missions have experienced technical issues late in the project lifecycle associated with the development and test of fault management capabilities, resulting in both project schedule delays and cost overruns. These issues are expected to cause increasing challenges as the spacecraft envisioned for future missions become more capable and complex. In recognition of the importance of addressing this problem, the Discovery and New Frontiers Program Office is planning a Fault Management Workshop, on behalf of NASA's Science Mission Directorate, Planetary Science Division, to bring together experts in fault management from across NASA, DoD, and industry. This will be a three-day, multi-Center workshop to identify fault management lessons learned, best practices, and future opportunities for investments. The scope focuses on deep-space and planetary robotic missions, with full recognition of the relevance of, and subsequent benefit to, Earth-orbiting missions.

Workshop attendees will review recent mission experiences, work together to understand common issues, identify lessons learned and best practices for fault management, and explore emerging technologies in this field. The outcome of the workshop will be a white paper documenting the key findings, which will be made available to all organizations involved in current and future space mission development.

The NASA SMD/PSD Fault Management Workshop will focus on the following three Topics:

1. Fault Management (FM) Architectures.
2. FM Verification and Validation (V&V).
3. FM Development Practices and Processes.

We are looking for participation from members of the spacecraft community who have experience in designing FM architectures, testing FM on simulators/testbeds/flight hardware, and operating flight systems where FM has been exercised. We also are interested in having participants who are involved in establishing organizational practices and processes that have an impact on FM design and development.

Day 1 will lay the foundation for the participants to understand how FM is conceived/designed/implemented/operated at the participating organizations, including NASA Centers (APL, GSFC, JPL, and MSFC), DoD, and relevant industry partners. Presentations will consist of Case Studies of recent missions from each organization, focusing on the three Topics listed above.

Day 2 will consist of three separate Breakout Sessions, one dedicated to each Topic. Registrants will participate in working groups to identify issues, to capture lessons learned, and to enumerate best practices. Day 2 also initiates a Poster Session for technologists to present concepts and technology developments relevant to the future of FM. The Poster Session continues throughout the remainder of the workshop.

Day 3 will begin with presentations to all participants summarizing the findings from each Breakout Session, with the goal of drawing conclusions from what we have learned in the previous days' activities. A group discussion follows, to explore and capture the issues, lessons learned, and best practices that cut across the three Topic areas. The day concludes with presentations on future directions and opportunities from researchers in academia, government, and industry.

The attendance at the FM Workshop will be limited in order to promote an interactive environment. The number of registered participants is targeted at 50-60, with all participants taking an active role in presenting, contributing in a Breakout Session, and/or authoring a poster. Online registration will open in February 2008.

Program Committee

Program Chair:
Jim Adams, Deputy Director, Planetary Science Division, NASA HQ
Program Host:
Paul Gilbert, Manager, Discovery and New Frontiers Office, NASA MSFC
Workshop Director:
John M. McDougal, MSFC
Steering Committee:
George Cancro, APL
Chris P. Jones, JPL
John M. McDougal, MSFC
Steven S. Scott, GSFC
Workshop Technical Coordinator and Point of Contact:
Lorraine M. Fesq, JPL, 818-393-7224, Lorraine.M.Fesq@jpl.nasa.gov

Location

The NASA SMD/PSD Fault Management Workshop will be held in the heart of the French Quarter, at the Ritz Carlton Hotel, at 921 Canal Street in New Orleans, LA. For more information on the venue, visit the hotel website at
http://www.ritzcarlton.com/en/Properties/NewOrleans.

This workshop is open to US Persons as defined by 22 CFR 120.15, which includes US citizens and lawful permanent residents in the US.

B. Workshop Overview

To find ways to ameliorate the schedule, cost, and predictability issues observed during the implementation of FM for planetary missions, PSD directed the Discovery and New Frontiers Program Office (D&NF PO) to host a workshop to bring together FM experts from government, industry, and academia to discuss their experiences and offer their perspective on solving these issues. To ensure broad participation, a Steering Committee was formed with representatives from APL, GSFC, JPL and MSFC.

The approach taken to organize the workshop was to assemble key players in the spacecraft FM field across NASA, industry and other organizations to:

- Capture the current state of FM.
- Expose the challenges associated with engineering and operating FM systems.
- Identify and describe the issues underlying these challenges.
- Discuss and document best practices and lessons learned in FM.
- Explore promising state-of-the-art technology and methodology solutions to identify potential investment targets.

The scope of the workshop focused on deep-space and planetary robotic missions since the observed challenges had all occurred on deep-space missions. However, it was recognized that Earth-orbiting missions also suffered from similar symptoms, although perhaps to a lesser degree, and that there was sufficient overlap in FM architectures and V&V methodologies to warrant strong representation and participation from the EO community. The scope specifically did not include human-rated missions with the acknowledgement that these missions involved additional FM issues that typically are not required on purely robotic missions. However, members of the human spaceflight community did attend with the goal of understanding the issues uncovered during the workshop and looking for lessons learned and best practices that are relevant to their missions.

C. Workshop Goals, Format and Activities

The NASA SMD/PSD Fault Management Workshop was held over 3 days (April 14 - 16, 2008) in New Orleans, Louisiana. Attendance at the workshop was limited in order to promote an interactive environment. All attendees were expected to actively participate in workshop through presentations, posters, and/or active participation in the dialog during the breakout sessions. In total, the workshop had approximately 100 attendees from 31 various organizations across government, industry, and academia.

The primary goal of the workshop was to identify technology and/or process issues driving Fault Management cost growth and schedule growth in unmanned, autonomous spacecraft today. Participants were tasked with identifying best practices to address those issues, and opportunities for investment to mitigate or resolve those issues. The expected outcome of the workshop was not to produce a recipe or a set of standards. Instead, the goal was to rise above institutional preferences and evaluate the applicability, strengths, and weaknesses associated with the different approaches.

The purpose for the workshop was to provide guidance for future programs and technology development. This guidance was to come in three major areas:

- Lessons Learned.
- Best Practices.
- Opportunities for investment.

The target audience for the workshop results and this white paper was FM practitioners, future proposal evaluators, reviewers, and project/program managers. It should be noted that the workshop was not looking to produce a recipe or a set of standards. Instead, the goal was to rise above institutional preferences and identify the strengths and weaknesses associated with different approaches, providing practitioners with the information required to knowledgeably tailor architectures and processes to their application.

The workshop was organized around four components: case studies, invited speakers, targeted round table discussions, and poster presentations. Figure A1 shows the workshop agenda, including case study presentations and invited speakers. Current FM approaches and techniques were collected using two formats. Participants in the workshop were requested to provide white paper inputs (RFWI) summarizing the history of fault management on their projects in addition to their participation in the workshop itself. In addition, 12 cases studies covering a variety of mission types and FM issues were selected for presentation at the workshop. Case study summaries, RFWI input, and abstracts for the Poster Presentations are included in the other Appendices.

The first day of the workshop was geared towards understanding the current state of fault management through case study presentations. Thirteen case studies were presented that included lessons learned from eight current or past missions. In addition to these lessons learned, presentations were also given on the current state of fault management practices and processes across various organizations.

For the second day of the workshop, the participants broke into three breakout sessions to focus in on detailed topics. The three breakout sessions focused on V&V, architectures, and practices, processes, and tools. Participants were asked to attend only a single breakout session over the course of the day, but were allowed to choose whichever breakout session most interested them. Participation was spread approximately equally across the three breakout sessions. Each breakout session was charged with identifying and characterizing cardinal issues in their particular sub-topic. The Poster Session began on the afternoon of Day 2, and continued as an adjunct activity throughout the rest of the workshop.

The third day was designed to capture the results of the workshop, to integrate the findings, and to present new ways to view the field of Fault Management. Each of the three Session Chairs from Day 2 presented a summary of findings from the three Breakout Sessions, stressing Lessons Learned, Best Practices, and Opportunities for Investment. The invited speakers from academia presented different perspectives and new insights into alternate approaches to FM software architectures and V&V techniques. The workshop ended with a group discussion on the final afternoon where each participant was given the opportunity to bring up any issues that were not already covered, summarize findings, or make any additional comments. It should be noted that each and every participant had something additional to add even after three days of discussions – showcasing the continuing spirit of open discussions and great dialog that was present throughout the entire workshop.

April 13, 2008
6pm - 8pm: Registration & Pre-event Reception

CDT	Day 1 - April 14, 2008
7:00	Registration
8:00	Welcome/Introductions/Logistics
8:15	Why Are We Here? - Jim Adams, HQ
8:30	Scope/Goals of the Workshop - Lorraine Fesq, JPL
9:00	urrent State: Case Studies from Recent Missio
9:15	"MER FLASH Anomaly" - Glenn Reeves, JPL
9:45	"Dawn Lessons Learned" - Jonathan Rustick, OSC
10:15	"TRMM Power System FM Case Study" - Kris Naylor, CSC
10:45	BREAK
10:55	"Case Study Results/Findings from SMC Flight Software Projects" - Douglas Buettner, Aerospac
11:15	"MRO In-Flight Articulation Anomaly" - Wayne Sidney, LM; Tim Halbrook, LM; Todd Bayer, JPL
Noon	Lunch - "Spacecraft Fault Management, a Historical Perspective" - invited speaker Gentry Lee, JPL
1:00	"FM Organizational and Process Issues" - Stephen Johnson, MSFC
1:30	"Hubble Space Telescope FM Lessons Learned" - Brian Vreeland, Space Systems Integration
2:00	"JWST Architecture and Processes" - Judy Tillman, NGST
2:30	"STEREO AHEAD IMU Anomaly" - Mike Trela, APL
2:50	BREAK
3:00	"Overview of FM Concepts Used on Typical Ball Spacecraft" - Christian Meyer, Ball Aerospace
3:30	"New Horizons Autonomy -- Path to Launch" - Adrian Hill, APL
4:00	assini Leaking Regulator Anomaly" - Brad Burt, J
4:30	Breakout Session Descriptions - Session Chairs
5:00	Steering Committee Tag-up
5:15	Breakout Session sub-committee meetings

CDT	Day 2 - April 15, 2008		
8:00	Parallel Break-out Sessions: Logistics		
8:15	Architectures Session Chair - George Cancro, APL	V&V Session co-Chairs - Ray Whitley, GSFC & Chris Jones, JPL	Practices, Processes, Tools Session Chair - Dave Watson, APL
9:45	BREAK + Poster Setup		
10:00	Architectures Session	V&V Session	Practices, Processes, Tools Session
11:30	Lunch		Session Chairs Meeting
12:30	Architectures Session	V&V Session	Practices, Processes, Tools Session
2:00	BREAK + Poster Session		
2:15	Architectures Session	V&V Session	Practices, Processes, Tools Session
3:30	Future Directions: "Improving System Qualit through Software Architecture" - invited speaker David Garlan, CMU		
4:30	Poster Session		Steering Committee Meeting
5:15			

CDT	Day 3 - April 16, 2008	
7:15	Poster Session	
8:15	Future Directions: "New Directions in V&V: Evidence, Arguments, and Automation" - invited speaker John Rushby, SRI	
9:15	Session Report - Architectures	
9:45	Session Report - V&V	
10:15	Session Report - Practices, Processes, Tools	
10:45	BREAK + Poster Session	
11:00	Future Directions: "Model-based Monitoring of Complex Systems" - invited speaker Brian Willams, MIT	
Noon	Lunch	Steering Committee Meeting
1:00	Group Discussion - Integration of Findings	
3:00	Closing Remarks - Adjourn	
3:15	Steering Committee and Session Chairs Meeting	
3:45		

Room Legend
French Quarter Bar - 3rd floor
Salon 3 - Ballroom Level
Salon 1 - Ballroom Level
Crescent View - 12th floor
Acadia - Ballroom Level
Broadmoor - Level 1
Salon Foyer (Posters & Breaks)

Figure A1. Fault Management Workshop agenda.

The workshop was well attended by 100 representatives from a variety of government, industry, and academic institutions (see Table A1). The attendees also brought expertise derived from a wide spectrum of missions, both in terms of operations, duration, and size, and functional roles.

Table A1. Participants and Missions represented at the SMD/PSD FM Workshop.

Institutions	
NASA	MSFC, JPL, JSC, ARC, HQ, GSFC, Stennis Space Center, JHU/APL
Other Government	AFRL, NRL, DARPA, NRESS
Academia	MIT, SRI, Carnegie Mellon University, Iowa State University,
Industry	Northrop Grumman Space Technology, Lockheed-Martin, Draper Laboratory, The Boeing
Missions	
Low Earth Orbit	HST, Chandra, TRMM
Deep-Space Missions	Cassini, New Horizons, MRO, Dawn, MER, JWST, DI, STEREO, Messenger
Other	TacSat, SDO, GPM, Constellation – Ares, Orion
Functional Roles	
Engineers	Software Reliability, spacecraft systems, software, technical supervisors, computer
Managers	Program managers, V&V managers, flight system managers, section heads, group
Academia	Program director, professors

VI Appendix B: Case Study Presentations

One of the primary goals of the workshop was to expose the current state of fault management. This goal was addressed on the first day of the workshop through a series of case study presentations. Thirteen case studies were presented that exposed issues dealing with in-flight anomalies, project FM flight experiences, project FM development experiences, and industry FM philosophy and approaches, as well as lessons learned from eight current or past missions. The following paragraphs summarize the case study presentations and identify common themes. The specifics of the presentations are being withheld to honor the closed nature of the workshop, which allowed presenters and participants to be quite candid and open.

A common message expressed in numerous presentations was that in many missions there existed indications of problems prior to an anomaly occurring, but the evidence was not examined or recognized in advance, perhaps due to incomplete testing, insufficient testbed fidelity, or lack of schedule to examine all of the test data. This is true both for test programs and in-flight data. In several cases, reviewing test results after an anomaly showed indications of the problem, but the problem was not identified in the test because the pass criteria of the test was met and schedule and budget pressures forced the test program forward. It is necessary to investigate all anomalies during a test program, and not simply determine if the test has passed the official pass/fail criteria. Test programs need to be created to test against all aspects of expected performance, not simply to test to find complete failures. Problems can be identified if the expected performance of the system does not match predictions in any area, including areas that are not being focused on in that particular test, and including performance that does not fail criteria but is also unexpected. In addition to indications of problems being missed in test programs, the indications of problems were also present, but not identified, in data during flight. While missions have telemetry sets with thousands of parameters, the parameters needed to diagnose a problem before it becomes an anomaly were often not available in telemetry. This is often due to a lack of depth in the telemetry. In a very similar situation to test programs, telemetry often only alerts the ground once an anomaly occurs, but does not provide insight to the ground when the system is performing in an unpredicted way that does not officially fail the pass/fail criteria. In both test programs and in telemetry definitions, we need to be looking for unexpected performance and early indicators of problems, instead of waiting until the thresholds for a given monitor are passed. Proper visibility into the behavior of these systems is key. Tools might need to be developed to more rapidly and accurately assess the data provided.

One of the root causes for not seeing indications of problems ahead of time is the complexity level of the fault-management systems being designed and used today. One of the large enablers to increased complexity in fault management systems is increased flexibility. It became clear in discussions that flexibility in systems is both a friend and a foe to the fault management engineer. From a sustaining engineering point of view, it was clear that flexibility in a fault management system is a very positive thing. The ability to modify the fault protection on the fly saved several missions from mission failure. However, from the test perspective, it was also clear that flexibility in a system can be very problematic. The response of a complex and flexible system can be very sensitive to the initial conditions and timing, making the system “fragile” and difficult to test. Even without additional complications from timing issues, N! permutations of a flexible system lead to N! test cases; to adequately test complex behavior, the number of test cases increases exponentially. When the additional sensitivities to timing get added in (timing of faults can affect response paths), the test program quickly becomes much too vast to cover every test exhaustively. This has led multiple missions to restrict operations only to states that have been exhaustively tested. If a change in configuration (i.e., a swap to the redundant side) is required in-flight, the mission is put on hold until the tests related to that change can be completed. It is clear that having a system that is capable of being flexible is important, but managing and restricting the flexibility to be used only when required is equally important.

Whether the system is designed to be flexible or not, it is clear that as mission concepts and spacecraft become more and more complex, the fault management systems associated with those spacecraft need to become more complex. As the systems become more complex, it is becoming increasingly important that the process of designing the fault management system needs to begin in the very early stages of the mission design process, and possibly requires new architectural approaches. The change needs to begin at the proposal, or initial planning level, and a strong fault management lead and team need to be supported throughout the program. Ground support equipment and personnel need to be brought onto the project early enough to make a difference. The message that fault management cannot be pushed back until the later stages of design was reiterated throughout many of the case study presentations. A suggested reason for the current situation is the notion that most, if not all, organizations do not recognize FM as a separate discipline, but rather as an adjunct duty for systems engineers or as part of the flight software development effort.

The biggest source of evidence that fault management is not considered early enough in the design process is the large increases between planned labor and actual labor hours in recent missions. One case study gave an example of the fault management task being planned for 0.5 FTE throughout the mission; in actuality, FM staffing peaked at 14 FTE during the V&V stages. This was a common occurrence. Evidence suggests that fault management is viewed initially as a side responsibility of a systems engineer, increased to a full time job as the mission progresses, and eventually an entire team is required to deal with problems, testing, etc. This is a problem not only from a budget stand point, but also logistically. People cannot start on a program and understand everything from the day they begin. Therefore, in these situations the core team needs to spend time not only getting their job done, but also training the new people who are needed to handle the much larger work load than was anticipated. The first and most obvious suggestion for improving this situation is to plan for an adequate work load based on past staffing metrics from the initial planning stages, including enough personnel and time for test program preparation and execution. In addition, the steady evolution of a FM architecture could improve the situation through re-use of test plans and the distribution of a knowledge base among personnel. Lessons learned from one project are easiest to apply to the next project if the personnel are the same. It is important to examine heritage carefully. There exist inherent differences between unique deep-space probes and similarly repeated Earth orbiting missions. Finally, a fault protection process should be defined at each organization with team definitions and responsibilities laid out from the early planning stages.

While it is clear that a team responsible for the fault management process needs to be defined and utilized early, it is not clear what the organizational home of that team should be. While workshop participants agreed that fault management is a systems-level issue, there was some disagreement about whether or not the team in charge of fault management should be fault management specialists or systems engineers focused on fault management. Having a large core team of systems engineers, without any specific fault management engineers, means that fault management will at times be pushed off for a higher priority systems issue. However, this team set-up would also imply that the engineers designing the fault management system would by definition be experts in the overall system and could better understand the interactions and needs of the fault management system. On the other hand, if fault management becomes a separate product team, the responsibility is more focused, which is good. However, making FM a separate product separates it from the design of the overall system, which makes the design process more difficult and might produce a less robust fault-protection system. A third option is to allow fault management to remain among the systems engineer responsibilities, but to have a person or sub-team specifically responsible for fault management. In this case, there could also be an integrated systems team, consisting of existing members of the systems engineering team focused on various aspects of the design (fault management, flight software, testing, etc.) that meets on a regular basis to be sure that everything is integrating correctly. No matter where the fault management team is organizationally located, the fact that the team needs to be pulled together at an earlier stage and that the analysts (FMEAs, PRA) and the designers (fault management, testing) need to have more interaction was universally agreed upon.

One of the main reasons that labor has been traditionally under budgeted for fault management systems is claims of heritage that are not accurate. In the proposal or planning stages many missions will claim or assume a large amount of heritage in the fault management system, even when it is clear that major modifications will be required. When something is classified as heritage or “re-use” it often only has to go through peer-review and not a full design review. This tempts management into calling a fault management system heritage even when it clear that major adaptations are required to meet the requirements of the new mission. Budgets are often set too low even for systems with actual re-use and heritage because it is assumed that the system will not require as extensive of a test program. Case studies have shown, however, that an extensive test program is still required for heritage systems. A new spacecraft with legacy software cannot be assumed to perform the same as a previous spacecraft, due to differences in timing and other minor system differences.

One of the reasons that a heritage system often will not work in a new mission is not technical, but cultural differences between partner institutions. Different organizations throughout the aerospace industry use different terminology and have varying fault management philosophies based on their organizational risk posture. Commercial organizations tend to have a slightly higher risk tolerance than government organizations. This is evident in their various fault management philosophies. This philosophy difference was stated as the difference between designing against probability versus designing against possibility. One specific example that was given was two ways to interpret the term “single fault tolerant.” In one organization’s definition, a design was able to survive any single fault, but that fault could be an operator error, a hardware error, a software error, etc. Any combination of those faults was considered beyond single fault and outside the scope of the fault management system. A second organization viewed single fault tolerant as any single fault, hardware or software, plus any set of errors, such as operational errors or SEU induced errors, which are not considered faults. There are even significant differences in the definitions of the terms used to describe the prevention and treatment of faults, such as fault management, fault protection, fault avoidance, or FDRIR (fault detection, response, isolation, and recovery). Representatives from various organizations are at times talking about the same thing and don’t know it, and at other times think they’re talking about the same thing and are actually talking about very different things. These differences became so clear that one of the major suggestions for future work that came out of the workshop was to develop a taxonomy for fault management, either at an institutional level or across the industry.

In addition to terminology differences across organizations causing difficulties in fault management design, terminology can also cause problems within the same organization. In some cases, multiple missions will use similar terminology, but with slightly different definitions. This can lead to confusion, misunderstandings, and errors in design. At least one major spacecraft anomaly could have been avoided if the terms used in the design had been strictly defined and clearly stated to all team members to avoid various interpretations.

While using the same terminology will help communications among team members, another communications issue that was brought up several times was an inability to visualize these complex systems. Fault management systems have become so complex that visualizing all the monitors, responses, and where everything is getting done is becoming very difficult. It is also extremely difficult to visualize how a change in the system (monitor, response, or parameter) will affect the rest of the system. While visualizing the system is difficult for everyone, it is especially difficult for systems engineers who are not software specialists and who, as discussed earlier, are often the ones designing the implementing the monitors and responses. It is important to develop and use tools, such as diagrams, that help with visualization. Telemetry visualization tools, that show what happened, in what order, and what state the telemetry is in (red, yellow, green) can also help operators to use these complex systems more efficiently.

One of the themes that ran throughout the case study presentations was the concept that fault management systems are required to be much more complex for deep-space missions than for Earth orbiting missions. Deep-space missions are generally long-life missions, with large periods of time in which contact with the ground is not possible. This greatly affects the fault management system. Deep-space probes have more responsibility placed upon the autonomy, due to their inability to rely upon ground operators for near/non-time sensitive anomalies. Since the on-board autonomy for deep-space missions must handle long periods of no contact, they must be more self-sufficient. In addition, many deep-space missions have time-critical events that cannot be avoided. This leads to a requirement for the fault management system to be fail-operational and not simply fail-safe during critical events. For both mission classes, however, it seems that health and safety are key, within the limitations of the redundancy and cross-strapping included with the hardware design. For any critical hardware subsystem, autonomous redundancy management is needed.

In addition to discussing issues that are common in the design process, the case study presentations also revealed several design guidelines that apply to all missions across the industry. The first of these design guidelines is to always provide a safety net, even if a specific fault has not been identified. Even if you haven’t identified an exact failure mechanism, you need to design a way for the system to fail safe in every situation. Even when failure modes have been identified, the knowledge of those failure modes will always be limited, and unexpected situations can occur. Therefore, it is important to put in rules that will protect the spacecraft from known dangerous situations, even if a fault has not been specifically identified that can lead to that exact situation. For all missions, the safing recovery procedures must be properly tested and practiced for rapid execution, if needed. The safing recovery procedures should include what data will be needed to identify and recover from an unknown fault and must, therefore, be recorded and stored.

While the complexity of future missions is driving the complexity of fault management systems, another design guideline that was mentioned several times was to keep the design as simple as possible and revert to fundamentals whenever possible. Although a high level of complexity is inherent in the FM of deep-space missions of this type, whenever possible, the simplest design is often the best. An example was given of a very complex analysis that was finessed to increase science return. In the end, the analysis was incorrect and a spacecraft anomaly occurred. Whenever dealing with extremely complex problems, it is a good idea to make simplifying assumptions and not forget the fundamentals of what the design is trying to achieve.

The final design guideline from the case study presentations was to be careful of the limitations of redundancy. Performance on a backup system could be worse than performance on the prime system, even with a minor fault in the prime system. In addition, calibrating the backup could cost time and/or other resources. Therefore, even when redundant systems are available, it is often preferable to remain on a prime system with a minor flaw. Consequently, autonomous changes to redundant systems should be treated with caution.

The Case Studies covered a number of key points that were discussed in the subsequent breakout session. The key points from these presentations are summarized below:

- FSW Development Problems, due to various factors, including inadequate requirements (insufficiently specified, incomplete, erroneous re-use application) — getting worse with complexity growth...
- Unforeseen impacts of faults...
- FP Architecture features which permit in-flight modification are invaluable...
- In-flight failures can be hard to predict - better to protect functionality...
- Limitations on redundancy.
- Flexible & Re-configurable FM is powerful.
- Flat architecture can be problematic; unexpected interactions.
- Scope of testing is daunting; tough to properly scope to get most bang for buck.
- Manage FP thru lifecycle; programmatic and SE approach.
- Build-in from bottom, KISS.
- Organizational and Process issues lead to problems with FM systems.
- Due to separation between analysts and designers (not as much of factor at JPL due to FP SE position seen as 'cross-disciplinary' — and they often do the FTA, FMEA themselves)...
- FM has no organizational home in NASA; not recognized as a discipline.
- FM Architecture Flexibility enables operators to modify.
- Use High Fidelity testbeds, with easy fault injection capability (often opposed notions).
- Test products with realistic initialization prior to upload.
- Improve Visibility Tools.

- FM Scope is more than just on-board FSW...
- Need FM Engineer throughout lifecycle...assign early in program.
- Include Flight System robustness to operator error (industry stance differs from customer stance).
- ITL Definition for FM Scenario tests is tough to establish and complete.
- Varying Test initial conditions is very important.
- Flexibility is a friend and foe.
- Flat architectures do not up scope well — lead to emergent behaviors.
- Visibility for diagnosis is important. Tools to quickly assess the data.
- Visualization tools are needed to see FM execution behavior.
- Architecture features to support quick FM changes is important.

In summary, the take-away themes captured during the Case Study Presentations are listed below.

1. FM Behavioral/Design Flaws not always detected during testing effort. Better tools to analyze test results will help catch problems which are not readily obvious. May be related to organizational issues.
2. Flexibility — A friend and foe...
3. Consider FM SE earlier in development schedule.
4. FM SE staffing not properly budgeted or scoped.
5. Misapplication / bad assumptions about heritage.
6. Terminology Confusion.
7. FM Complexity differences between Earth orbiters and Deep Space.
8. Common Design Guidelines — Safety Net, Develop/Test Safing Recovery Procedure, KISS, redundancy limitations.

VII Appendix C: Breakout Sessions

Day two of the Workshop was dedicated to a discussion session to enable the participants to discuss the issues presented in the Case Studies on the previous day and to suggest additional issues that were relevant to the Workshop. The goal of the Breakout Sessions was to distill the information to uncover the root causes of the issues. The end products were to identify lessons learned, best practices, and opportunities for investment that would benefit future missions when designing FM systems.

The Breakout Sessions were organized around three focus areas: FM Architectures, FM V&V, and FM Practices, Processes and Tools. The following subsections describe an overview of each of the three Sessions.

A. Architectures Breakout Session

The Fault Management Architectures Breakout Session was chaired by George Cancro from JHU/APL, with support from Adrian Hill from JHU/APL, Kevin Barltrop and Michel Ingham from JPL, and Tim Crumbley from MSFC.

1. Scope/Goals

The goal of the Architecture session were as follows:

- To develop the principles, requirements, and features of a good fault management architecture for unmanned deep-space vehicles though surveying the current state-of-practice to understand current flaws and essential features.
- To develop the appropriate roles of hardware and software, including functional and physical redundancy.
- Understanding the effect of architectures on process, testing, verification, and operations.

The Fault Management Architecture breakout session group included leading experts from NASA and industry in the area of spacecraft FM Architecture. Lessons learned were gathered through a process of group discussions and small focus groups in the areas of FM architectures state of the practice, principles of good FM architectures, H/W and S/W interactions, and the effect of FM architectures on process, V&V, and Operations.

2. Focus Areas and Discussion Questions

The following focus areas and general questions were posed to motivate discussion in the breakout session:

1. **State of the Practice** — Collect and document current state-of-practice across industry and NASA. Try to develop a "family tree" of systems.
   - What requirements have driven your FM Architecture to be as it is today?
   - What are the essential features of your current FM Architecture?
   - What is the biggest flaw in your current FM Architecture?

2. **Features, Requirements and Principles for FM Architectures** — Collect and document the features, requirements and principles that should be core to a good FM Architecture.
   - What basic principles and/or requirements should all FM architectures have?
   - What are the features that Operators desire in FM Architectures?
   - What are the features that Testers desire in FM Architectures?
3. **H/W and S/W interaction** — Discuss and Document the appropriate roles of H/W and S/W as well as how redundancy should be used.
   - What are the roles of hardware and software in your architecture?
   - What should be used to drive your hardware-software allocation?
   - What is the appropriate use of physical and functional redundancy?
4. **Effects on Process, V&V and Ops** — Discuss and Document how FM Architectures affect these areas and how these areas effect FM Architectures, both beneficially and adversely.
   - Can you name any beneficial impact that FM Architectures have had on Process, V&V or Ops?
   - How has FM architecture effected testing in the past?
   - Can process and architecture be more closely coupled?

3. General Observations on FM Architectures

The implementation of FM architectures within the software domain is very similar across NASA, JHU/APL, and industry. Just as requirements tend to follow certain themes, fault management software architectures are quite similar across NASA, JPL, APL, and industry. In general, software architectures follow the natural division between fault detection and fault response. Within both detection and response, projects have implemented systems that can be easily categorized.

FM architectures can be easily categorized using six basic concepts: fault detection can be either distributed or central; fault response is either distributed or central and either parallel or serial. In practice, though the language describing architectures varies, most systems appear to employ all of these methods at some level.

Fault detection methods tend to be very similar. Telemetry is collected from assemblies and subsystem functions and compared against an acceptable value range to determine if the assembly or function is behaving as expected. To filter for transient data values and false detections, persistence schemes are implemented that check that a telemetry point is consistently in violation for some reasonable period.

Fault detection might be implemented by a central monitor that executes independently of the rest of the system – sometimes even as a separate software object, and in one architecture, on a separate processor. A more common approach appears to be distributing fault detection throughout the system: local hardware or low-level software performs checks on telemetry at the collection point and reports faults to the appropriate handler.

Fault responses are usually simple, self-contained command blocks that address a single fault. They are implemented either as simple serial command blocks or a logic chart with some conditional execution. In some cases, for extra flexibility, a logic chart is used to determine the command block to run, but only the simple command block is used to configure the system.

Fault response can be categorized as distributed or central, similar to detection. Responses are sometimes implemented locally, executing within the hardware or software object that detects the fault. Local responses are useful when a fault response requires fast turnaround, but because distributed response also means that responses are run in parallel without any arbitration, distributed architectures can result in extra complexity and unexpected emergent behavior. More frequently, responses are managed by a central executor that might allow parallel execution of command blocks, but that can also be used to limit responses to serial execution for simplicity. The central executor also provides the benefit of prioritization of responses, and arbitration between responses if two responses require conflicting system resources.

The majority of architectures in practice appear to use a combination of these concepts. Fault detection at the component level is usually distributed in the hardware or component-level software, along the lines of the system's functional decomposition, while system-level fault detection is performed in higher, more central parts of the system. Some systems will use local responses for limited reconfigurations or masking, but most of the time the local detector will report the fault to a central executor that will collect the reported faults and manage execution of responses. Responses are command blocks that will execute in serial or limited parallel fashion. None of the information provided suggests an architecture that allows unlimited parallel execution of faults, whether as a limitation of the system or as a design choice.

One extreme example of a mixed architecture implemented two separate executors that share fault management responsibilities, but each system follows a distributed detection/central response scheme. One response engine uses parallel response execution because of the criticality of completing responses quickly, while the other limits to serial.

One additional complication to the FM architecture stems from the common need to fail operational: re-entrant critical events. Should a fault require the system to halt execution of a critical event, the system will need to have the capability to resume execution after the fault has been addressed. This basic concept arose in a few responses and was implied in others, but appears to be, like the other elements of FM architectures, a common problem.

B. Fault Management Verification and Validation

The FM V&V Session was chaired by Chris Jones from JPL and Ray Whitley from GSFC, with support from Mike Trela and Brian Bauer from JHU/APL, Cathy White from MSFC, Manuel Maldonado and Dave Everett from GSFC, and Arden Acord from JPL.

1. Scope/Goals

This FM V&V Session addressed the current state of verification and validation of fault management systems. Methods, completeness (coverage), escape prevention, and the role of V&V in the development process were examined. Specifically, the relationships between V&V

and FM architecture and FM design processes were discussed. An assessment of current practices was made, and suggestions for future approaches were described.

In the realm of FM, the process of V&V is the tool used to flesh out the probable fault areas, particularly those that are not apparent from the design process. This process should be one not simply where the ability to meet requirements in a single instant is demonstrated, but rather where the ability to prevent events from disabling the system from meeting the requirements (i.e., how can we break the system through our testing process?).

Lessons learned for Verification and Validation were gathered through a process of group discussions with representatives from NASA, DoD, and industry.

For clarity in terminology, the following definitions are required.

- **Verification**: *Demonstration that an end product generated from product implementation or product integration conforms to its design solution definition requirements as a function of the product-line life-cycle phase and the location of the WBS model end product in the system structure.* **(NPR 7123.1A)** ["Did you build the product right?]
- **Validation**: *Confirmation that a verified end product generated by product implementation or product integration fulfills (satisfies) its intended use when placed in its intended environment and to ensure that any anomalies discovered during validation are appropriately resolved prior to delivery of the product (if validation is done by the supplier of the product) or prior to integration with other products into a higher-level assembled product (if validation is done by the receiver of the product). The validation is done against the set of baselined stakeholder expectations*. **(NPR 7123.1A)** ["Did you build the right Product?"]

From a fault management perspective, the V&V process serves to ensure compliance with requirements and to bring out potential faults as early as possible in the system development process.

2. Focus Areas and Discussion Questions

- What does our in-flight experience with fault management tell us about current V&V methodologies?
- Actual in-flight behavior (commanded/degraded) was not anticipated.
- Tripped up by untested (or untestable) features?
- Limited by testbed fidelity?
- Do we really "test as we fly?"
- What does our system V&V experience tell us about our architectures, design, design capture methodologies and project lifecycle?
- Limited test coverage due to broad design space or Earth-based limitations.
- Architecture created a "forest" of trees or lack of coherent architecture.
- Poor documentation of design prevented thorough test program.
- Late deliveries and hardware/software incompatibilities.

- What are the V&V drivers?
- Design complexity.
- System performance requirements.
- Fail-op requirements.
- COTS hardware/software, etc.
- How do modeling and simulation support FM V&V?
- Are the technologies adequate?
- What is the desired future state?
- What is the appropriate scope of a FM V&V program?
- Test coverage.
- Incompressible test list make-up.
- Hardware/software, component/subsystem/system, simple fault scenarios/stress testing.
- How do we know when we're through?

C. FM Development Practices, Processes & Tools (P/P/T)

Dave Watson (JHU/APL) chaired the FM P/P/T Session, with support from Brad Burt (JPL), Bob Henderson (JHU/APL), and Jonathan Mead (Northrop Grumman).

1. Scope/Goals

The Practices, Processes and Tools breakout session aimed to address the issues associated with tools and development methodologies or processes in the design and development of spacecraft fault management (FM) subsystems. Such subsystems, by their nature, present a particular challenge in development as they interact with all other spacecraft subsystems and are dependent on them during overall system development. The role of fault management designers and developers on spacecraft development teams, the processes they follow, and the tools they use was a particular emphasis of this session. The focus was on what organizational constructs and processes might have been particularly successful or problematic in the past and what tools have proven successful in analysis of FM requirements, developing FM logic, and then implementing and testing that logic.

For the purpose of the discussion, the definition of "tools" ranged from early analysis methodologies, such as event sequence diagrams and fault tree analysis, through FMEA and traditional requirements allocation systems into development frameworks and implementation languages.

2. Focus Areas and Discussion Questions

A set of general questions were posed to motivate discussion in the breakout session:

1. What tools or processes are used to estimate the cost and schedule for FM subsystem analysis, design and development? What are the metrics for describing FM complexity?

2. What design tools are traditionally used in the different phases of FM analysis and development?
3. What tools are currently under investigation at your organization? What are the evaluation criteria and experiences with these potential tools?
4. Would it be feasible to link fault analysis, fault management design, testing, implementation, and maintenance through a tightly integrated tool set?
5. To what extent should system and software engineering process be driven by available tools?
6. Do standard development process specifications address FM in particular? What are the benefits and limitations of current process specifications to FM development?
7. Are particular organizational constructs beneficial or problematic in the development of FM subsystems?
8. Are there particularly beneficial or problematic aspects of mission operations interaction with onboard FM?
9. What are the implications of FM complexity/simplicity that factor into mission operations (risk and cost)? Should FM logic provide defense against errors introduced in operations?
10. What are the reusable (mission-independent) aspects of FM process and design?

3. Session Attendance

The session was well attended, with 33 individuals representing a broad spectrum of NASA, civilian, and military space domains. Represented organizations were as follows:

- NASA HQ, JPL, MSFC, ARC.
- Naval Research Lab.
- Lockheed Martin.
- Orbital Sciences.
- Northrop Grumman.
- Interface and Control Systems.
- Aerospace Corporation.
- Draper Lab.
- General Atomics.
- Ball Aerospace.

VIII Appendix D: Invited Speakers

During the workshop, we invited three speakers from academia to present to the FM community. The purpose of these presentations was to provide a different perspective and some insight into future directions. We invited researchers from academia who were leaders in the fields most relevant to the focus areas of this workshop: namely, software architecture, V&V, and a unified approach to FM designs.

The first day included a lunchtime presentation from Gentry Lee on the historical perspective of fault management systems and what we should be striving for in the future.

On Day 2, Dr. David Garlan, Professor of Computer Science and Director of the Software Engineering Professional Programs at Carnegie Mellon University, gave a talk entitled "Improving System Quality through Software Architecture." Dr. Garlan is considered to be one of the founders of the field of software architecture and, in particular, formal representation and analysis of architectural designs. Over the past decade and a half, there has been increasing understanding about the role that software architecture can and should play in mastering the complexity of complex software system design, providing a basis for early analysis and prediction, ensuring that systems retain their structural integrity over time, and enabling re-use and dramatic cost reductions. In this talk, Dr. Garlan outlined some of the key insights that drive the field and identified some of the salient features of software architecture as they relate to improving the dependability of software-based systems, focusing on techniques to (a) express architectural descriptions precisely and unambiguously; (b) provide soundness criteria and tools to check consistency of architectural designs; (c) analyze those designs to determine implied system properties; (d) exploit patterns and styles, and check whether a given architecture conforms to a given pattern; (e) guarantee that the implementation of a system is consistent with its architectural design; and (f) support self-healing capabilities.

With respect to fault management, Dr. Garlan identified a number of specific ways in which a more disciplined approach to software architecture in NASA might help:

- Improve Communication: Architecture helps make design commitments explicit; architectural specifications can make that intent precise.
- Reduce System Complexity: Done properly, architecture designs would address FM as top-level architectural concern using standard, uniform, and comprehensive principles. It would ensure that fault management was part of the essential design process (not added later as a separate effort). It would also open to discussion the fundamental nature of FM – for example, questioning the common belief/practice that FM should not be considered as part the normal control behavior of the system.
- Improved V&V: The use of architecture-based analyses would allow NASA engineers to reason about FM coverage, completeness, correctness, timing, subsystem interaction. It could also help ensure conformance in code through generation, refinement or conformance analysis.
- Reduce Brittleness: Architectures can be used to design FM architecture that is resilient to change. Complex interactions can be done early using formal models and model checking tools.

- Reduce Cost: Architecture can be used as a basis for:
  - Generating code automatically from models.
  - Codifying and reusing architectural principles that have proven integrity.
  - Knowing the limitations of re-use.
  - Amortizing design efforts over families of systems.

While these potential benefits of a disciplined and effective approach to software architecture could have a significant impact on FM, currently there are a number of institutional impediments that make it difficult for NASA to embrace these techniques. These include:

- Inappropriate modeling techniques: For example, in many project at NASA the view is that "Software architecture is just boxes and lines," "Software architecture is just code modules," "A layered diagram says it all."
- Misunderstanding about role of architecture in product lines and architectural re-use: For example, today many view product lines as simply a re-use library, failing to recognize the essential role that architectures play in process.
- An impoverished culture of architecture design: Some symptoms of this are: (a) weak standards for architectural description and analysis; (b) lack of institutionalized architectural reviews early in the design process when they can be most productive; (c) lack of a common vocabulary of architecture and expectations about what an architecture should accomplish; and (d) a lack of a solid architecture education among engineers.

On Day 3, Dr. John Rushby, Program Director for Formal Methods and Dependable Systems within the Computer Science Laboratory at SRI International, spoke of "New Directions in V&V: Evidence, Arguments, and Automation." Dr. Rushby's research interests center on the use of formal methods for problems in the design and assurance of secure and dependable systems. In his talk, Dr. Rushby explained that the central problem in V&V is to anticipate all possible scenarios that can arise in operation of the system and to establish that it behaves appropriately in every one of them. When the system is composed of interacting subsystems (such as a spacecraft), the number of possible scenarios is exponential in the number of components, and when continuous variables (such as time) are involved, it can become unbounded.

Testing and simulation can examine only a fraction of such a huge number of scenarios. Dr. Rushby described how automated techniques based on formal methods and model-based descriptions can allow all possible scenarios to be considered and he explained their relationship to traditional methods such as FTA and FMEA. He also described how automation can be extended to test case generation and runtime monitoring. These new methods differ substantially from standards-based approaches to assurance such as DO-178B and are best employed in a goal-based approach that makes arguments and evidence explicit.

Our third invited speaker from academia was Dr. Brian C. Williams. Prof. Williams is a Professor of Aeronautics and Astronautics at the Massachusetts Institute of Technology, as well as the Director of the Autonomous Systems Laboratory (ASL) and a member of the Computer Science and Artificial Intelligence Laboratory (CSAIL). Prof. Williams' research concentrates on model-based monitoring and fault management and on model-based autonomy: the creation of long-lived autonomous systems that are able to explore, command, diagnose, and repair themselves by employing online reasoning. This work has been employed to develop a wide range of fault robust and autonomous systems, including deep-space probes, automobiles, copiers, naval ships, autonomous air, ground and undersea vehicles, and walking robots.

Similar to the problem of Verification and Validation addressed in Dr. Rushby's talk, the fault-protection systems engineer has the challenge of anticipating all possible fault scenarios that can arise in operation of the system and to establish fault responses that behave appropriately in every one of them. The number of possible scenarios is again at best exponential in the number of components.

Professor Williams' talk, entitled "Model-Based Monitoring of Complex Systems," confronted this challenge through four parts. The first part of the talk focused on NASA's trend towards developing increasingly ambitious missions, which need to operate through failure without moving into a 'safing' mode. This requires software systems that are able to maintain function despite several types of disturbances and failures, by employing a range of recovery mechanisms, such as dynamic scheduling, hardware reconfiguration, and hybrid control. Achieving the desired level of fault robustness in a cost-effective manner requires new programming languages, architectures, and design practices. Professor Williams introduced one such programming paradigm, called *Model-based Programming*, and a corresponding language, called RMPL. In this approach, a programmer writes a *control program*, which specifies a system's intended behavior in terms of constraints on the systems' state over time (e.g., "Engine A thrusting in 2 hrs"), and a *plant model*, which specifies the behavior of the system being controlled. An RMPL program executive then employs the plant model 1) to map the state specification provided by the control program to control actions, 2) to confirm, based on sensor information, that the intended states are achieved, and 3) to diagnose and recover when failures occur. In the talk, example RMPL programs were demonstrated on two different types of systems: a spacecraft and a humanoid robot.

In the second part of the talk, Professor Williams provided a tutorial on the foundations of the model-based diagnostic approach. Model-based diagnosis performs systems wide diagnosis based on a model of a system and observations from the system's sensors. The model is a modular description of a system, such as a hardware schematic and a set of component models. These component models include descriptions of both correct and faulty modes of behavior, but do not presume complete knowledge of a component's failure modes. Given a set of observations, model-based diagnosis performs a system-wide diagnosis that identifies symptomatic behavior, identifies faulty components, proposes likely failure modes for these faulty components, and identifies components that are failing in a novel manner. Features of model-based diagnosis include diagnosing multiple faults, un-modeled fault behaviors, sensor faults, intermittent faults, and faults with delayed symptoms. Likewise, model-based methods exist for automatically generating hardware reconfigurations and repair.

In the third part of the talk, Professor Williams examined rule-based fault management architectures that have been employed in a range of missions, such as Cassini, NEAR, and Messenger. These architectures have proven quite effective in the past; however, for current and future missions, mission complexity has resulted in increasingly large rule sets that collectively are becoming difficult to analyze and debug. In this part of the talk, he discussed how model-based diagnosis algorithms (the MiniMe system) have been adapted to automatically generate diagnostic rules, which can be executed in a manner similar to previous rule engines, but avoids the problems of rule interactions of these earlier approaches.

In the fourth and final part of the talk, Professor Williams described the current state of the art in model-based methods for runtime monitoring and diagnosis of systems with complex behaviors. One suite of methods takes as input models of both software and hardware components and uses them to detect symptoms manifest in the software, as well as the hardware, and to isolate both software and hardware failures. Software models are specified in a hierarchical automata-like language in the spirit of State Charts, a language that is frequently employed by systems engineers. A second suite of methods takes as input component models that include a mix of discrete and continuous behaviors in the form of hybrid models that combines discrete stochastic automata and ordinary differential equations. These hybrid monitoring and diagnosis methods can be used during prognosis to detect the onset of failure or to detect incipient failures, such as small leaks. A final suite of *active control methods* were presented that improve a system's ability to isolate failures by placing the system in states in which sensor information disambiguates between likely failure models, while ensuring that system function is maintained.

IX Appendix E: Poster Session

A. Poster Session Summary

One of the primary goals of this workshop was to explore emerging technologies and discuss future opportunities for investments to improve fault management for future missions. With this in mind, a poster session was held for the final two days of the three day workshop. The poster session provided an opportunity for technologists to present concepts and technology developments relevant to the future of FM. In total, 13 posters were presented. The titles and authors of all the posters are shown in Table E1 and the abstracts for all of the posters are provided in Section B of this Appendix. In general, the posters focused on a few major areas: V&V, reliability, flight software and fault protection architectures, and case studies.

Table E1: Poster titles and authors.

Poster Title	Author(s)
Automatic Testcase Generation for Flight Software	D. Bushnell (RIACS/NASA ARC),
Criticality-Based Design of Fault Management Software	D. Scheidt (JHU APL)
Design and Validation of Robust Fault Tolerant Systems: Markov Chains Combined with Dynamic Modeling	R. Bradshaw (Draper Laboratory), J. Zinchuk (Draper Laboratory)
ExecSpec: Visually Designing and Operating a Finite State Machine-based Spacecraft Autonomy System	R. Turner (JHU APL), S. Hooda (JHU APL), J. Gersh (JHU APL), G. Cancro (JHU APL)
Fault Management for an Autonomous Robotic Satellite Grappling Mission	J. Lennon (NRL)
Fault Protection in a Goal-Based Control Architecture	D. Dvorak (JPL)
Implementation of Integrated System Fault Management Capability	F. Figueroa (NASA SSC), J. Schmalzel (NASA SSC), J. Morris (Jacobs Technology), H. Smith (Jacobs Technology),
Intelligent Systems Health Monitoring for Autonomous Lunar Exploration: A Case Study	J. Frank (NASA ARC)
Run-Time Flight Software Modification for Fault Management	A. Murray (JPL)
The Nemesis Research Project	K. Barltrop (JPL)
Timeliner Architecture for Fault Management	R. Barrington (Draper Laboratory)
Tool-Supported Verification of Contingency Design	R. Lutz (JPL/Caltech & ISU), A. Patterson-Hine (NASA ARC)
Verification of Model-based Software Systems for Space Applications	G. Brat (USRA RIACS)

Several of the posters focused on future directions of V&V for fault management. USRA RIACS and JPL have partnered together to develop technology for automatic testcase generation for flight software. The enabling technologies for the approaches presented are model checking and symbolic execution. Model checking is an automated technique for software verification.

Symbolic execution evaluates programs with symbolic values and represents variable values as symbolic expressions.

JPL is working on a generalized framework for systems validation that can be applied to both traditional and autonomous systems. The Nemesis research project is developing the framework, which consists of an automated test case generation and execution system that rapidly and thoroughly identifies flaws or vulnerabilities within a system, using a combination of genetic algorithm optimization, goal-seeking algorithms, and war games. The project should result in earlier and more complete identification of flaws compared to current approaches.

JPL, ISU, and NASA ARC have partnered together to develop a technique to perform early investigation of anomalous behaviors and early, integrated modeling of the software system and its faults. The technique uses model-based verification of the contingency design and assists developers in identifying and remedying design gaps. Tool-supported verification of contingency design facilitates fault management by helping reduce fault-related re-works during testing and by making the design more robust to operational contingencies.

USRA RIACS presented a poster focused on advanced V&V techniques for model-based software, especially planning systems. The systems are broken into two distinct parts – the domain model and the planning model. The planning engine can be verified in terms of its mechanisms using automatic proving techniques, and only needs to be validated once. The domain models capture knowledge specific to a given problem and each new domain model needs to be validated separately.

In the area of reliability analysis, Draper Laboratory has developed a new tool set for dynamic reliability analyses using a Markov based modeling approach. The new software tool, “PARADyM” uses a capability to dynamically model system behavior while enumerating potential fault combinations and their resulting effects on the system. PARADyM provides estimated reliability as well as a sensitivity analysis showing weak aspects of the system.

Two posters focused on the development of new fault protection architectures. NASA SSC and Jacobs Technology are working on a comprehensive approach to implement integrated fault management. This effort encompasses maturing several necessary technologies including intelligent sensors and software environments. A set of tools and objects have been developed to allow users to model systems for integrated fault management. These models can represent every element in a system of systems as an intelligent element, including process models that define relationships among elements.

JHU APL is developing “intelligent control” systems to devise a fault management strategy at run-time. This approach examines the comparative utility of various fault management algorithms to determine algorithm effectiveness as a function of system complexity and dynamics. This analysis can then be used to select optimal algorithms for a given scenario, and can be performed autonomously at run-time. This allows for the design and construction of a hybrid fault management subsystem that switches among various algorithms to maintain optimal fault response.

Several additional posters were presented on flight software architectures that will enhance fault protection systems. JHU APL is exploring a new visual programming approach to autonomy development based on a combination of finite state machines and data flow diagrams to form an executable specification. This platform allows users and developers to construct reusable diagrams of state machines and link them together to form autonomy components and subsystems in a visual environment. The platform has the capability to upload a specification of the finite state machine directly to the spacecraft at any time during development or operations. Additional visualization tools on the ground allow the user to display the spacecraft behavior by highlighting portions of the state-transition diagram.

Draper Laboratory has developed a scripting utility that can provide a flexible capability to respond to changes in performance and configuration. The Timeliner sequencer allows for automation of re-configuration and/or re-direction of objectives after an unexpected failure. Timeliner was recently utilized on the DARPA Orbital Express (OE) program to perform the Mission Management function, control the subsystem functions, monitor for failures, and cause appropriate contingency responses to execute.

JPL is working on an object-oriented and component-based flight software design that is based in C++. The design provides the capability to replace or add components of the flight software at run-time, without having to reboot the system. This allows for easy uplink and installation of new FSW logic, including updates to fault protection and/or entirely new fault protection logic. This would be especially useful in the event of a change in the system behavior due to a fault or a change in the environment that was not anticipated.

In addition to pure flight software architectures or pure fault management architectures, one poster focused on a new architecture that integrates flight software and fault management into a single, unified architecture. JPL has developed a goal- and state-based control architecture that handles execution of nominal activities and fault protection. Given a scheduled plan expressed in the form of a "goal network" and alternative execution options captured in "goal elaboration tactics", the control system monitors the execution of goals and decides on what actions to take if a goal is not met. This architecture simplifies design and verification because it uses the same mechanisms for nominal control as it does for fault protection.

Finally, two posters focused on case-studies of particularly challenging future missions or simulations of future missions. NRL is working on the Front-End Robotics Enabling Near-Term Demonstration (FREND) program, which requires the autonomous grapple of a simulated geostationary satellite. This mission has two major challenges from a fault protection standpoint. The first challenge is events which would or would not be considered faults at different points in the mission. The second challenge is that critical autonomous tasks, such as grappling, are not compatible with classic fault handling, such as safing the spacecraft.

NASA ARC has conducted a software simulation of an autonomous lunar lander. For this simulated mission, Intelligent Systems Health Monitoring (ISHM) was in integral part of the software design. JPL's Spacecraft Health Inference Engine (SHINE) was used in a fault detection role, and NASA ARC's Hybrid Diagnosis Engine (HyDE) was used for fault isolation. The control software was integrated with a physics simulation of lunar surface operations, and successfully demonstrated detection of, and recovery from, a variety of faults.

B. Poster Session Abstracts

Automatic Testcase Generation for Flight Software

David Bushnell, RIACS/NASA Ames Research Center

Corina Pasareanu, Perot Systems/NASA Ames Research Center

Ryan MacKey, Jet Propulsion Laboratory

The TacSat3 project is applying Integrated Systems Health Management (ISHM) technologies to an Air Force spacecraft for operational evaluation in space. The experiment will demonstrate the effectiveness and cost of ISHM and vehicle systems management (VSM) technologies through onboard operation for extended periods.

We present two approaches to automatic testcase generation for ISHM:

1. A blackbox approach that views the system as a blackbox, and uses a grammar-based specification of the system's inputs to automatically generate *all* inputs that satisfy the specifications (up to pre-specified limits); these inputs are then used to exercise the system.
2. A whitebox approach that performs analysis and testcase generation directly on a representation of the internal behaviour of the system under test.

The enabling technologies for both these approaches are model checking and symbolic execution, as implemented in the Ames' Java Path Finder (JPF) tool suite.

Model checking is an automated technique for software verification. Unlike simulation and testing which check only some of the system executions and therefore may miss errors, model checking exhaustively explores all possible executions. Symbolic execution evaluates programs with symbolic rather than concrete values and represents variable values as symbolic expressions.

We are applying the blackbox approach to generating input scripts for the Spacecraft Command Language (SCL) from Interface and Control Systems. SCL is an embedded interpreter for controlling spacecraft systems. TacSat3 will be using SCL as the controller for its ISHM systems.

We translated the SCL grammar into a program that outputs scripts conforming to the grammars. Running JPF on this program generates all legal input scripts up to a prespecified size. Script generation can also be targeted to specific parts of the grammar of interest to the developers. These scripts are then fed to the SCL Executive. ICS's in-house coverage tools will be run to measure code coverage. Because the scripts exercise all parts of the grammar, we expect them to provide high code coverage. This blackbox approach is suitable for systems for which we do not have access to the source code.

We are applying whitebox test generation to the Spacecraft Health INference Engine (SHINE) that is part of the ISHM system. In TacSat3, SHINE will execute an on-board knowledge base for fault detection and diagnosis. SHINE converts its knowledge base into optimized C code which runs onboard TacSat3.

SHINE can translate its rules into an intermediate representation (Java) suitable for analysis with JPF. JPF will analyze SHINE's Java output using symbolic execution, producing testcases that can provide either complete or directed coverage of the code.

Automatically generated test suites can provide full code coverage and be quickly regenerated when code changes. Because our tools analyze executable code, they fully cover the delivered code, not just models of the code.

This approach also provides a way to generate tests that exercise specific sections of code under specific preconditions. This capability gives us more focused testing of specific sections of code.

Criticality-Based Design of Fault Management Software

David Scheidt, JHU Applied Physics Laboratory

As a system, spacecraft are complex due to the large number of feasible system states that could be produced by one or more faults. Spacecraft complexity prevents both engineers and automated design tools from generating optimal stimulus-response rules for all feasible failure scenarios during design. This limitation requires fault management designers to choose between two of the attributes: correctness, completeness and timeliness, but not all three. Fault management systems currently in use on APL spacecraft are designed to be correct and timely, with spacecraft safing being used as a default response to unanticipated failure combinations. An alternative approach to fault management that has been used by NASA-Ames on DS-1 and APL on the Air Force MSX spacecraft is the use of "intelligent control". Intelligent control uses an implicit or explicit system model to devise a fault management strategy at run-time. Soft computing, a general category of algorithms that includes fuzzy logic, neural networks and some forms of Bayesian belief networks are a well studied alternative to fault management. Selecting the appropriate fault management algorithm is an art, rather than a science. The effort described in this paper seeks to provide first-principles understanding of the comparative utility of various fault management algorithms, as well as providing insight into the optimal design of hybrid architectures and tailoring of fault management systems. The chief insight discussed is the recognition that algorithm effectiveness varies as a function of system complexity and dynamics. Further, this effectiveness may be expressed as an efficiency curve across these domains. The effectiveness curves from candidate algorithms may be overlaid to create a Pareto optimal surface which can, in turn, be used to select the optimal algorithms for a given scenario. We begin our discussion by defining metrics for fault management in terms of mission criticality. Mission criticality combines the probability that the spacecraft can satisfy a priori defined mission requirements with the time-sensitivity of the fault response. Formalisms for system complexity and system dynamics are then defined. These formalisms combine complexity theory and information theory, a key element of which is *entropic drag*, which is the rate at which information is lost do to unpredictable dynamics in a system. We show how reducing approximation error through high-fidelity representation can cause a net loss of knowledge and controllability due to the impact of entropic drag. We show how an analysis of system complexity and dynamics can be applied to algorithm performance models to select the optimal representational fidelity and optimal fault management algorithm. This analysis can be performed autonomously at run-time, allowing for the design and construction of a hybrid fault management subsystem that switches representational fidelity and algorithm to maintain optimal fault response. Finally we introduce criticality-based reasoning, a technique for designing intelligent control algorithms that uses mission-criticality as the primary fitness criteria by which solutions are examined.

Design and Validation of Robust Fault Tolerant Systems: Markov Chains Combined with Dynamic Modeling

Rich Bradshaw, Draper Laboratory

Jeff Zinchuk, Draper Laboratory

Fault tolerant design and fault management has become common place practice for deep-space and human space flight missions where poor or non-existent fault detection and containment can result in loss of mission, loss of mission critical hardware or even loss of human life. Using a Markov based modeling approach, Draper Labs has developed a new tool set for the development and testing of fault tolerant systems. The new software tool, “PARADyM”, brings forth a new capability to dynamically model system behavior while enumerating potential fault combinations and their resulting effects on the system. Using this information PARADyM provides the estimated reliability for the intended mission as well as a sensitivity analysis showing weak or overbuilt aspects of the system. This presentation will provide an overview of the Markov based approach in combination with an interactive demo of the PARADyM tool.

ExecSpec: Visually Designing and Operating a Finite State Machine-based Spacecraft Autonomy System

Russell Turner (JHU Applied Physics Laboratory)

Sharjeel Hooda (JHU Applied Physics Laboratory)

John Gersh (JHU Applied Physics Laboratory)

George Cancro (JHU Applied Physics Laboratory)

The increasing complexity of modern spacecraft autonomy systems makes it difficult for them to be completely understood by engineers, mission operators and domain experts during design and development as well as testing and flight operations. To address these problems, The Johns Hopkins University Applied Physics Laboratory is exploring a new visual programming approach to autonomy development based on a combination of finite state machines and data flow diagrams to form an executable specification. This allows autonomy developers to interactively construct reusable diagrams of state machines and link them together to form autonomy components and subsystems. This presentation describes our current work towards creating such a visual autonomy development environment, which we call ExecSpec. We provide an overview of the system, show how we address the visual complexity caused by multiple state machine systems, and show how mission operators could use such visual monitoring tools during flight to analyze onboard autonomy status.

Finite state machine state-transition diagrams have been used on previous spacecraft autonomy projects; however, these have been translated into autonomy flight code by automatic code generators or by software developers, which removes flexibility for future change and severely limits re-use. In contrast, the ExecSpec development environment can output a specification of the finite state machine system itself which can be uploaded directly to the spacecraft at any time during development or operations. Once on-board, a generic FSM interpreter executes the specification and reports the status of its execution through telemetry to the visual tool on the ground, which displays the spacecraft behavior by highlighting portions of the state-transition diagram. In this way, ExecSpec serves as a visual tool allowing the developer to design, test, debug, and deploy, as well as monitor and modify the behavior, of a spacecraft autonomy system all within a single context.

In addition, ExecSpec can support the rapid assembly of autonomy systems through a prototype-instance methodology: individual finite state machines are encapsulated with well-defined input and output variables that can be interconnected to form networks of interacting finite state machines. These visual system components can be saved as libraries of component prototypes that can be used to instantiate and assemble new autonomy system diagrams; thus, increasing reusability of autonomy systems between spacecrafts and decreasing total development time.

Fault Management for an Autonomous Robotic Satellite Grappling Mission

Jamie Lennon, Naval Research Laboratory

The Front-End Robotics Enabling Near-Term Demonstration (FREND) program requires the autonomous grapple of a simulated geostationary satellite. From a fault management perspective, we have two major challenges. First, events which are faults during some phases of the mission are nominal events in others. We address this through the use of a finite state machine-based Mission Sequencer which interprets flags sent by subsystem monitors based on the current state of the system. Second, lessons learned from the recently concluded Orbital Express mission included the observation that traditional spacecraft fault handling (e.g., system safing whenever any off-nominal event is detected) greatly complicates critical autonomous tasks, such as grappling or refueling. We are continuing to investigate system operations possibilities that will balance safety and mission performance, including multiple attempts to visually identify our grappling feature while in proximity to our customer spacecraft.

Fault Protection in a Goal-Based Control Architecture

Daniel L. Dvorak, Jet Propulsion Laboratory

Historically, fault protection has been designed as a separate, event-driven system that monitors conditions in a spacecraft and responds autonomously, taking over control from the nominal system. In effect, two control architectures are engineered to work together: an open-loop time-based command sequencer for nominal activities, and a closed-loop event-driven control system for fault detection and response. This bifurcated approach has worked, more or less, but has always been difficult to engineer. One reason for the difficulty has been due to poor separation of concerns in design, as typified in threshold-and-persistency monitors used for triggering fault responses. Such monitors convolve state estimation with control, making it hard to tune the parameters of all such monitors to achieve the desired response in every situation. Another reason for the difficulty has been in the growing pressure to localize fault responses and thereby avoid disruption to unaffected activities.

This poster shows how fault protection can be handled in a single, unified control architecture for nominal activities and fault protection, both of which are controlled by “goals” and “goal elaborators”. In this architecture, a goal is constraint on the value of a state variable over a time interval, and can therefore represent control objectives (e.g. slew the spacecraft to within epsilon of a given vector), dependencies (e.g. this device must be healthy), and flight rules (e.g. an instrument’s temperature must remain in a specified range). Every type of goal has an associated goal elaborator that produces, at planning time, a set of supporting goals (if any) needed for its achievement. After planning, the entire collection of goals, including parent-child relationships and temporal ordering dependencies, is organized in a goal network. During execution, each goal’s constraint is actively monitored, and if the constraint is ever violated — whether due to a fault or to unanticipated conditions — the goal failure is reported to the elaborator that produced that goal. The elaborator can make a choice, in the context of the parent goal’s objective and the current system state, to continue on a best effort basis, or respond with a new elaboration, or escalate the problem to its parent, or immediately invoke safing.

This goal-based control architecture cleanly separates state estimation from control decisions and situates fault protection decisions within the context of the activities that are affected. Further, this architecture simplifies design and verification in that it uses the same mechanisms for nominal control as for fault protection. The net result is a control architecture that not only dispenses with the bifurcated approach to fault protection but also naturally allows for increasing levels of autonomy by virtue of its closed-loop control paradigm.

Implementation of Integrated System Fault Management Capability

Fernando Figueroa, NASA Stennis Space Center

John Schmalzel, NASA Stennis Space Center

Jon Morris, Jacobs Technology (NASA SSC)

Harvey Smith, Jacobs Technology (NASA SSC)

Mark Turowski, Jacobs Technology (NASA SSC)

Fault management can be focused on different goals. For example, on evaluating readiness of a system for a mission, or on diagnosing causes of anomalies, or predicting future anomalies. The focus of fault management (FM) at NASA Stennis has been to determine system health.

FM is viewed as a capability that incorporates the following functions: (1) detection of faults, (2) diagnosis of causes, (3) determination of effects throughout a system, (4) prediction of future system anomalies, (4) Assessment of system health, and (5) user interfaces for integrated awareness by the operator(s) and users. However, our work has emphasized that all functions must reflect integrated approaches that exploit cause-effect relationships among elements in systems-of-systems (SoS).

Significant effort has been devoted to mature technology areas needed for implementation of credible FM capability. These areas include: (1) architecture, taxonomy, and ontology based on a definition of a SoS as a hierarchical network of intelligent elements; (2) a software environment to implement Integrated System Fault Management, including tools to automate root-cause analysis; (3) intelligent sensors; (4) anomaly detection algorithms, approaches, strategies, and methodologies; (5) standards for interoperability and plug&play related to management of data, information, and knowledge (DIaK); and (6) user interfaces for integrated awareness. Based on these efforts, prototype implementations have been developed in operational testbeds.

Intelligent sensors (physical and virtual) must have the ability to determine their own health (just as any intelligent element in the system must). In general, assessing fault condition is distributed throughout all system elements. Sensors deserve special attention, because often the most significant question asked when doing any type of inquiry on a system is if the sensors providing the information are believable. FM on sensors is especially critical, as they are crucial in the operation of any modern system with feedback control. We have approached sensor FM by considering a sensor to be a system itself. It uses local data, information, and knowledge (DIaK) to assess and manage faults; but it also accepts DIaK from the rest of the system to help it improve its fault assessment.

The poster will describe a comprehensive approach to implement integrated FM, which is applicable to ground and space settings. The software environment used for FM is based on G2, which is a commercial product that includes tools geared for development of applications with embedded intelligence (object oriented compartmentalization and management of DIaK). G2 includes an inference engine, network capability, gateways for communicating with products that adhere to standards (e.g. SQL, Allen Bradley programmable Logic Controllers, etc), and an engine for implementation of root-cause analysis. A set of tools and objects have been developed to make possible the building of models of systems for integrated fault management. These models can represent every element in a SoS as an intelligent element, incorporate information and knowledge about each element, including process models that define relationships among elements, which are used for consistency checks that lead to detection of anomalies (failures). Generic root-cause trees have also been implemented for typical faults on sensors, valves, and other elements. These trees are re-usable like the majority of the object classes used to create the SoS models for integrated fault management. The software environment inherently supports evolution of the capability as more and better algorithms, models, etc. for FM become available.

Intelligent Systems Health Monitoring for Autonomous Lunar Exploration: A Case Study

Jeremy Frank, NASA Ames Research Center

We conducted a short-duration, medium fidelity software simulation of an autonomous lunar lander designed for science missions. The mission design assumed the spacecraft would move on the lunar surface using the same software and hardware employed for descent and landing.

While high degrees of on-board automation have proven themselves in a variety of recent flight experiments, such software is still considered a risk in many space missions. For this reason we chose a "separation architecture", in which we distinguished between traditional flight software functions (referred to as the Reactive Layer) and higher-order functions making use of automation technologies like Planners, Executives, and intelligent systems health monitoring (referred to as the Deliberative Layer).

Intelligent System Health Monitoring (ISHM) software was an integral part of the software design. We employed the Jet Propulsion Laboratory's Spacecraft Health Inference Engine (SHINE) system in the Reactive Layer, and NASA Ames Research Center's Hybrid Diagnosis Engine (HyDE) in the Deliberative Layer. SHINE performs high-speed logical inference, and acts primarily in a Fault Detection (FD) role, while HyDE performs fault isolation using combinatorial search. In addition to these functions, each software component has self-test functions invoked by the Executive to detect software faults. Finally, SHINE was tasked to detect software faults corresponding to failures of the Deliberative Layer software.

We integrated the control software with a medium-fidelity physics simulation of lunar surface operations, and successfully demonstrated detection of a variety of Main Engine and Attitude Control System faults, and detect and recover from simulated failures of the Executive. We also designed experiments to detect and recover from failures of one of the solar panels. Among our lessons learned, we describe experiences in system integration, software integration, and experiences in integrating multiple models.

Run-Time Flight Software Modification for Fault Management

Alex Murray, Jet Propulsion Laboratory

This poster presents an Object-oriented and Component-based Flight Software (FSW) design that was developed in C++ for the instrument flight software of the Aquarius mission, along with a discussion of the advantages of this design for fault management. The Component-based design provides the capability to replace Components of the FSW at run time, or to add entirely new FSW Components at run time, without having to reboot the system. In terms of fault management, this capability provides a number of significant advantages: it allows easy uplink and installation of new FSW logic to perform additional monitoring of suspect hardware, software, or environment conditions, in support of diagnosis of a problem or suspected problem, or even for purposes of fault anticipation and prevention. New active fault protection logic can be easily installed. The capability to handle new commands or produce new telemetry can be easily installed. Being able to do this with no reboot is a tremendous operational advantage, because it can take a complex flight system many hours of operations time to get back to mission mode after a reboot. In addition, the ability to automatically recover from a suspended thread without rebooting, provided by this design, could make real-time fault protection in time-critical mission activities more robust.

The Nemesis Research Project

Kevin Barltrop, Jet Propulsion Laboratory

Autonomous systems are more difficult to validate than traditional systems because they require more numerous and complex behaviors in order to operate self-sufficiently. The Nemesis research project was initiated to develop a generalized framework for systems validation that can be applied to both traditional and autonomous systems. The framework consists of an automated test case generation and execution system that rapidly and thoroughly identifies flaws or vulnerabilities within a system. By applying genetic optimization and goal-seeking algorithms on the test equipment side, we conduct a "war game" between an autonomous system and its complementary nemesis. The end result of the war games is a collection of scenarios that reveal any undesirable behaviors of the system under test.

The near-term research consists of two phases: A proof-of-concept phase in which a reusable framework for Nemesis is developed that consists of a goal-seeking module, a genetic algorithm model, and templates for application-specific elements; a calibration phase in which Nemesis is applied to a legacy, non-goal seeking system (with a catalog of known test results) to demonstrate its utility and effectiveness in comparison to traditional system validation methods. The preliminary results demonstrated an increased likely of finding significant system flaws early in the test campaign compared to random selection of tests. When augmented with engineering expertise to seed the search, we expect to see more earlier and more complete identification of flaws compare to the current approaches.

Timeliner Architecture for Fault Management

Ray Barrington, Draper Laboratory

If built into the spacecraft command and data architecture, a scripting utility onboard a spacecraft can provide a flexible capability to respond to changes in performance and configuration. Observed performance is space may be different than tested or may change over time requiring compensation or limit updates. Unexpected failures require re-configuration and/or re-direction of objectives. A scripting capability allows for automation of re-configuration and re-direction that is easy to modify from the ground as operators gain mission experience or objectives change. Recently, Draper's Timeliner sequencer was utilized on the DARPA Orbital Express (OE) program to perform the Mission Management function as well as control of subsystem functions. For OE, Timeliner was extended to include a monitor that was used to alert the Mission Manager of failures and cause appropriate contingency response to execute. This Timeliner tool as well as an architecture scared for scripted commanding, proved to be extremely useful in responding to changing conditions and unexpected performance during the OE mission.

Tool-Supported Verification of Contingency Design

Robyn Lutz, JPL/Caltech & ISU

Ann Patterson-Hine, NASA Ames Research Center

Advances in autonomy will allow future NASA missions to be robust to a much wider range of anomalies than ever before. Such systems will go beyond traditional fault protection to anticipate, identify and handle additional environmental or operational scenarios that might add risk. These broader classes of anomalies are called contingencies.

This poster describes a technique that we have developed to perform early investigation of contingencies (anomalous behaviors) and early, integrated modeling of the software system and its faults. The technique uses model-based verification to evaluate whether the software design provides coverage of the contingency-related software requirements. This tool-supported verification of the contingency design assists developers in identifying and remedying design gaps and in demonstrating coverage using automatically generated metrics. The tool support used was the TEAMS tool from QSI, currently in use at several NASA centers.

Tool-supported verification of contingency design facilitates fault management by helping reduce costly, unexpected, fault-related re-works during testing and by making the design more robust to operational contingencies. Model-based development often postpones fault modeling until after the functional behavior of the system has been specified. This poster demonstrates the advantages of instead performing early investigation of contingencies (anomalous behaviors) and early, integrated modeling of the software system and its faults. Software tool support such as this will be an asset in handling the scale, increased autonomy and evolving software requirements of future, envisioned NASA missions.

Illustrative examples from three NASA applications (an unpiloted aerial vehicle, MER critical pointing software, and a spacecraft electrical power system testbed) show how early consideration of potential anomalies helps build in robustness. Previous results from this work have been presented in three conferences/workshops and one journal, and will be surveyed and updated in this poster.

Some of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration, and at NASA Ames Research Center, and was funded by NASA's Office of Safety and Mission Assurance Software Assurance Research Program, administered at JPL by ATPO.

Verification of Model-based Software Systems for Space Applications

Guillaume Brat, USRA Research Institute for Advanced Computer Science

Model-based software can play an important role in future space applications. For example, it can help streamline ground operations, or, assist in autonomous rendezvous and docking operations. Moreover, model-based software, especially planning and scheduling systems, are well suited to finding solutions to recovery problems. For example, planners can be used to explore the space of recovery actions for a power subsystem and implement a solution without (or with minimal) human intervention. In general, the exploration capabilities of model-based systems give them great flexibility. Unfortunately, it also makes them unpredictable to our human eyes, both in terms of their execution and their verification, hence, the necessity of gaining confidence in the safety of these systems through advanced V&V techniques.

Planning systems are made of two parts: the domain model describes the domain on which the planner can reason and the planning engine performs the reasoning (usually in the form of a systematic exploration of the state space induced by the planning goals and the domain model). These two parts yield different V&V challenges. The planning engine can be verified in terms of its mechanisms, i.e., check that forward propagation is done correctly, check that constraints are elaborated correctly, and so on. We believe this can be done using automatic proving techniques. The use of these techniques comes with a high cost, but the planning engine only needs to be validated once. In some sense, it is a bit similar to validating a compiler. Domain models change depending on the applications and therefore require different V&V techniques.

Domain models capture knowledge specific to a given problem. Each new domain model needs to be validated. In mission critical applications, it is necessary to guarantee that the executions of plans do not violate certain safety requirements (called flight rules). In previous work (published at MBT'07) we have shown how a flight rule can be encoded as an LTL formula *F*. Verification can either be performed on *F* directly by taking its negation and translating it to a PDDL planning goal, or a set of test cases can be generated from *F* using requirement-based testing techniques. This second method of verification is useful when developers are interested in the behavior of the domain with respect to some requirement; for instance, a requirement of the form "if the rover is moving, then all instruments are stowed" might be true because of the antecedent being always false (i.e. the rover never moves in the plan), and the simple verification of the property would not provide this information. On the contrary, a test for the proposition "the rover is moving" being true would fail, giving more details about the domain. Intuitively, the set of test cases is derived by applying a generation algorithm to *F*, which extends the standard notion of MC/DC coverage to requirements written in LTL; each generated test case is then characterized by a LTL formula, called the trap formula for the test case.

X Appendix F: Request for Workshop Input (RFWI) Responses

In addition to their participation in the workshop itself, participants were asked to provide written inputs about the history of fault management on their projects. Participants were provided with a questionnaire that covered many of the common challenges in fault management design, but were also encouraged to provide extra information they felt was relevant to a discussion of the state of the practice. Respondents provided a wealth of insight to their design practices, their architectures, and their verification and validation approaches.

A total of 16 responses were provided, categorized in Table F1. Eight were direct responses to the questionnaire. The others were previously published materials that addressed some or all of the subject areas covered by the questionnaire: some from specific missions and some general discussions of fault management design at an organization.

Table F1. Information provided in response to Request for Workshop Input.

Mission Type	RFWI Response	Other Materials
Earth Orbiter	2	1
Planetary Orbiter	2	3
Lander	1	
Flyby	3	2
Other General Input		2

The 16 inputs spanned 11 diverse missions, including three Earth orbiters, five planetary orbiters, one lander, and two fly-by encounter missions from practitioners in a variety of organizations. Some missions provided multiple inputs spanning the different organizations involved or additional previous papers written on their FM systems. In some cases, respondents also provided insight into practices in other branches of their organization that did not participate in the workshop. While the technical challenges faced by projects were diverse, approaches to fault management and the programmatic challenges of implementing fault management systems were similar in many ways: high-level requirements followed certain themes and the architectures that rose out of those requirements share many similar concepts.

Approaches, lessons learned, and best practices gleaned from the RFWI responses have been incorporated into the results of the workshop presented in the Findings and Recommendations sections of this document.